Universidad de Córdoba
Instituto de Estudios de Postgrado
Máster en Ingeniería Informática

Trabajo Fin de Máster:

Segmentación de series temporales mediante un algoritmo multiobjetivo evolutivo

Autor:
- Antonio Manuel Durán Rosal

Directores:
- César Hervás Martínez
- Pedro Antonio Gutiérrez Peña

Enero, 2016

Este Trabajo Fin de Máster ha sido financiado en parte con cargo al Proyecto **TIN2014-54583-C2-1-R** del Ministerio de Ciencia y Tecnología (MICYT), con fondos FEDER, con el Proyecto **P11-TIC-7508** de la Junta de Andalucía. La investigación de Antonio Manuel Durán Rosal está subvencionada por el programa predoctoral de Formación de Personal Universitario (FPU, referencia **FPU14/03039**) del Ministerio de Educación y Ciencia (MEC).

Segmentación de series temporales mediante un algoritmo multiobjetivo evolutivo

Firmado en Córdoba, Enero de 2016

Directores:

Fdo: Dr. César Hervás Martínez

Fdo: Dr. Pedro Antonio Gutiérrez Peña

Autor:

Fdo: Antonio Manuel Durán Rosal

César Hervás Martínez, Catedrático de Universidad del Dpto. de Informática y Análisis Numérico en la Escuela Politécnica Superior de la Universidad de Córdoba, investigador responsable del Grupo de Investigación AYRNA.

Informa:

Que el presente Trabajo Fin de Máster titulado *Segmentación de series temporales mediante un algoritmo multiobjetivo evolutivo*, constituye la memoria presentada por **Antonio Manuel Durán Rosal** para aspirar a la obtención del título correspondiente al Máster Universitario en Ingeniería Informática, y ha sido realizado bajo mi dirección en la Escuela Politécnica Superior de la Universidad de Córdoba reuniendo, a mi juicio, las condiciones necesarias exigidas en este tipo de trabajos.

Y para que conste, se expide y firma el presente certificado en Córdoba, Enero 2016.

El Director:

Fdo: Prof. Dr. César Hervás Martínez

Pedro Antonio Gutiérrez Peña, Profesor Contratado Doctor del Dpto. de Informática y Análisis Numérico en la Escuela Politécnica Superior de la Universidad de Córdoba e investigador del grupo AYRNA.

Informa:

Que el presente Trabajo Fin de Máster titulado *Segmentación de series temporales mediante un algoritmo multiobjetivo evolutivo*, constituye la memoria presentada por **Antonio Manuel Durán Rosal** para aspirar a la obtención del título correspondiente al Máster Universitario en Ingeniería Informática, y ha sido realizado bajo mi dirección en la Escuela Politécnica Superior de la Universidad de Córdoba reuniendo, a mi juicio, las condiciones necesarias exigidas en este tipo de trabajos.

Y para que conste, se expide y firma el presente certificado en Córdoba, Enero 2016.

El Director:

Fdo: Prof. Dr. Pedro Antonio Gutiérrez Peña

Agradecimientos

Mis agradecimientos van para mis padres y para mi familia en general, que siempre me han apoyado y motivado a progresar en mi trayectoria académica. Gracias también a Ana, por saber cómo convertir cada instante en inolvidable y desprender tanta felicidad.

También querría dar las gracias a todos los compañeros de AYRNA y del Máster que me han ayudado durante esta etapa, ya sea trabajando cooperativamente, como competitivamente, ya que sin ellos nada habría sido igual, no dudo que sabrán que son ellos cuando lean estas líneas.

Por supuesto agradecer a mis directores del Trabajo Fin de Máster, César y Pedro, sin los cuales este Trabajo Fin de Máster no habría sido posible, muchas gracias por toda la paciencia que han tenido conmigo y por su dedicación. Por último, me gustaría agradecer a los profesores que me han formado en la Universidad.

Índice General

Índice de Figuras

Índice de Tablas

Parte I

Memoria

1 Introducción

En este primer capítulo, se expondrá de manera resumida y general el problema abordado en el presente Trabajo Fin de Máster y los objetivos que se pretenden cumplir para la resolución de éste. Se analizará someramente la importancia de los algoritmos multiobjetivo evolutivos y de aprendizaje no supervisado en la actualidad y sus múltiples usos, de forma que justifiquemos la importancia del algoritmo que ha sido desarrollado en este Trabajo Fin de Máster.

Hoy en día, nos vemos inundados por una gran cantidad de datos, en diversas formas (bases de datos, series temporales, etc.). La cantidad de estos datos crece de una forma exponencial, por lo que se carece del tiempo necesario para analizarlos convenientemente. De esta forma, debemos encontrar mecanismos que nos ayuden a obtener información útil de manera automática a partir de esta gran cantidad de datos.

Centrándonos en el tema del Trabajo Fin de Máster, una serie temporal consiste en una secuencia de datos, observaciones o valores medidos en determinados momentos (equidistantes o no) y ordenados cronológicamente. Tratar de forma automática series temporales es un tema muy difícil debido a que la naturaleza de las mismas depende del ámbito de aplicación: series de temperatura, de cantidad de oxígeno, datos ecónomicos referidos a índices bursátiles, datos relacionados con energías renovables, como puede ser la velocidad de viento, etc. El

único aspecto en común es que presentan una función que varía con respecto al tiempo, pero la tipología de esta variación puede ser o no estacionaria, estar gobernada por diversas variables, presentar o no una componente aleatoria, etc.

El análisis de series temporales se puede plantear desde muchas perspectivas: representación en otros formatos (o compresión), **segmentación de series temporales**, comparación de series, predicción de valores futuros, clasificación de un conjunto de series en distintas categorías, *clustering* de series temporales, etc. Resulta difícil imaginar una rama de las ciencias en la que no aparezcan datos que puedan ser considerados cómo series temporales.

La segmentación de series temporales consiste en dividir las series temporales en distintos instantes de tiempo, con el fin de satisfacer unos objetivos determinados. En la actualidad, se tienen dos puntos de vista para dicho próposito.

Por un lado, el dividir las series en pequeñas subseries temporales ha sido un procedimiento para descubrir patrones de similaridad en distintos períodos de tiempo de las mismas. Para ello, es necesario dividir la serie y después realizar un *clustering* de los subsegmentos generados, y así observar sus similitudes. Relacionado con este punto de vista, el encontrar patrones de similaridad puede ayudarnos con la detección de anomalías de la serie, ya que, al agrupar trozos de la serie, se pueden observar cuales son los menos frecuentes, y así, poder analizar la naturaleza de los mismos. El último aspecto importante en esta perspectiva es la implementación de sistemas de alerta temprana para los *Tipping Points* o eventos críticos a lo largo de la serie, ya que analizando lo que ha ocurrido previamente a dichos eventos podremos hacer predicciones de estos eventos y así prepararnos para instantes futuros.

Por otro lado, la segmentación se puede llevar a cabo para encontrar un subconjunto de puntos característicos de la serie que aproximen lo mejor posible dicha serie sin perder una gran cantidad de información, o, lo que es lo mismo, minimizando al máximo el error cometido en

dicha aproximación. Cada segmento es aproximado por un modelo simple, de forma que la serie puede ser reconstruida incurriendo en el menor error posible. Esta perspectiva es de vital importancia para el análisis de grandes bases de datos temporales en tareas posteriores a la segmentación, para reducir el coste computacional.

Un análisis de la optimización de ambos objetivos nos hace considerar "a priori" que la determinación de puntos o segmentos específicos de la serie y la aproximación de la serie temporal en todo su dominio son objetivos contrapuestos y que, por tanto, hay que hacer un enfoque multiobjetivo asociado a su optimización conjunta. Esta claro que una segmentación con muchos segmentos incurrirá en un menor error de aproximación, pero dichos segmentos serán más difíciles de agrupar. Por el contrario, si el número de segmentos es reducido, será más fácil agruparlos pero el error de aproximación se incrementará.

Una solución para abordar problemas de segmentación es usar algoritmos evolutivos y en concreto, en nuestro caso, algoritmos genéticos. Esto algoritmos son métodos de búsqueda global. Este tipo de algoritmos incorporan la semántica de la evolución natural a los procesos de optimización, convirtiéndose en métodos estocásticos de búsqueda ciega de soluciones cuasióptimas. Trabajan manteniendo una población de individuos, en nuestro caso, un conjunto de series temporales segmentadas (con distintos puntos de corte). Éstas son sometidas a una serie de transformaciones: por un lado tenemos un operador de cruce dónde la combinación de unos individuos producen otros que comparten información genética de sus progenitores. Además, se tiene un operador de mutación dónde a partir de la modificación de un individuo se obtiene otro. La idea sería que los puntos de corte de la segmentación fuesen aproximados por el algoritmo evolutivo, buscando una codificación correcta de las soluciones.

Una variante muy conveniente de estos algoritmos para el problema que se pretende abordar en este Trabajo Fin de Máster, son los algoritmos multiobjetivo evolutivos. Como su nombre indica, estos algoritmos tienen una filosofía similar a los algoritmos evolutivos pero

el proceso de optimización consiste en hacerla considerando conjuntamente varios objetivos en lugar de uno sólo. Así pues, la aplicación de algoritmos multiobjetivo evolutivos, nos permitiría obtener soluciones que por un lado aproximan muy bien la serie con un subconjunto de puntos, y que por otro, poseen información de patrones contenidos en la serie con el mismo subconjunto de puntos.

Además del algoritmo multiobjetivo evolutivo, la segmentación debe incorporar un algoritmo de agrupamiento o *clustering* para agrupar cada uno de los segmentos, resultantes de la segmentación, en función de su similitud y así garantizar que existen patrones comunes a lo largo de la serie temporal. De esta forma, lograríamos, no solo obtener los puntos de corte, sino clasificar los segmentos en un conjunto predefinido de clases o categorías.

Este *clustering* se engloba dentro del denominado aprendizaje automático no supervisado, en el que los datos no se presentan etiquetados, sino que lo que se pretende es encontrar estos grupos y las características que diferencian a unos de otros, determinar la distribución de los datos, o proyectar los datos en un espacio de menor dimensión para visualizarlos.

Por ello, este Trabajo Fin de Máster se centrará en la segmentación de series temporales aplicando un algoritmo de *clustering* particional determinista (la inicialización del algoritmo es determinista independientemente de la semilla) incluido en el ámbito del aprendizaje automático no supervisado, además de un algoritmo multiobjetivo evolutivo para mejorar dicha búsqueda de la mejor solución, incluido en el ámbito de la computación evolutiva.

Para poder aplicar un algoritmo de *clustering*, es necesario caracterizar cada uno de los segmentos y luego agruparlos en función de estas características. Se han considerado una serie de características estadísticas y de tendencia de las subseries o segmentos que serán explicadas con más detalle en los siguientes capítulos.

La idea es hacer un algoritmo genérico que sirva de herramienta

para el análisis de cualquier tipo de serie temporal. De hecho, en este Trabajo, la metodología propuesta se aplicará a series temporales de distinta índole como son una serie sintética, una serie de altura de olas, otra de economía referida a un índice búrsatil como es el IBEX35, y una seríe relacionada con electrocardiogramas.

Resumiendo, los algoritmos multiobjetivo evolutivos y los algoritmos no supervisados de *clustering* son interesantes para extraer información de series temporales de distinta temática, además de conseguir una buena aproximación de la misma mediante un subconjunto de puntos, lo que nos ahorraría tiempo de cómputo para trabajos futuros, aunque este tipo de algoritmos pueden presentar muchos problemas e inconvenientes para obtener un método adecuado de segmentación.

1.1. Objetivos

Se pretende implementar un algoritmo multiobjetivo evolutivo con dos funciones de aptitud contrapuestas. La primera función de *fitness* estará relacionada con el objetivo de la similaridad de subseries según distintas características y será obtenida mediante un algoritmo de clustering particional determinista. Mientras que la segunda función de *fitness* estará relacionada con el objetivo de la aproximación de la serie y será implementada mediante distintas métricas de error, tales como la raíz cuadrada del error cuadrático medio (*Root Mean Square Error*, RMSE). Los objetivos propuestos para tal fin son los siguientes:

1. Realizar un estudio teórico sobre las temáticas del proyecto, incluyendo aprendizaje automático no supervisado (distintos métodos de *clustering*), los algoritmos multiobjetivo evolutivos y los algoritmos de segmentación de series temporales.

2. Implementar un primer algoritmo en el que sólo se tenga en cuenta como función de *fitness* la relacionada con el error de aproximación de la serie.

3. Modificar el algoritmo propuesto previamente y publicado en la revista *Climate Dynamics* [Nikolaou et al., 2014][1], en el que se realiza una segmentación basada únicamente en encontrar patrones similares, sistemas de alerta temprana y *Tipping Points*.

4. Basándonos en las ideas propuestas en los dos algoritmos anteriores, implementar el algoritmo multiobjetivo perseguido en este Trabajo Fin de Máster.

5. Evaluar el algoritmo con distintas bases de datos de diferente naturaleza o ámbito. Esto incluye dos subobjetivos:

   a) Adaptar las series temporales a un formato común.

   b) Estudiar una configuración común para todos los experimentos y que funcione bien en todas las bases de datos.

6. Servir de apoyo a futuras investigaciones acerca de la predicción de series temporales a partir de una segmentación dada.

7. Escribir trabajos científicos donde se propongan estas ideas, algoritmos y metodologías, y presentarlos a revistas y/o congresos de ámbito internacional que tengan una alta posibilidad de divulgación científica.

[1] Algoritmo desarrollado durante el Trabajo Fin de Grado del autor de este Trabajo Fin de Máster.

2 Antecedentes

En este capítulo se pretende concretar el punto de partida del Trabajo Fin de Máster. Este conocimiento base nos permitirá especificar, en los siguientes capítulos, el sistema a desarrollar. A continuación, se mostrará una breve revisión u *overview* acerca de la segmentación de series temporales, y la explicación detallada de los algoritmos más importantes del Estado del Arte en los que se basa nuestro Trabajo Fin de Máster.

2.1. *Overview* de la Segmentación de Series Temporales

Recientemente, la importancia de los datos organizados de forma temporal ha supuesto un amplio estudio e investigación de las series temporales en el campo de la minería de datos. En este sentido, las series temporales aparecen en cualquier rama de la ciencia. La minería de datos aplicada a series temporales comprende un conjunto de tareas muy diverso, entre los que destaca la segmentación de series temporales, que, cómo se ha explicado previamente, consiste en la división de la serie temporal en distintos instantes de tiempo con el fin

de satisfacer unos objetivos determinados.

Aplicar un algoritmo de segmentación para encontrar patrones de similaridad ya ha sido investigado por algunos autores, entre los que destacan los trabajos de Chung et al. [2004] o Tseng et al. [2009], en los que los autores hacen uso de algoritmos evolutivos. Por otro lado, Das et al. [1998] incluye una ventana fija para resolver el problema de la segmentación, y así reducir la serie a una sucesión de segmentos simples. Relacionado con este punto de vista, la caracterización de eventos extraños o segmentos especiales de la serie (conocidos como *Tipping Points* [Nikolaou et al., 2014]) pueden ser usados para el descubrimiento de segmentos previos comunes que ocurren antes de que se produzcan dichos eventos y así poder extraer conclusiones acerca de su naturaleza y servir como sistema de alerta temprana. Finalmente, la búsqueda de segmentos "raros" es uno de los caminos para identificar anomalías en la serie temporal [Fuchs et al., 2009].

Bajo otro punto de vista, simplificar la serie temporal mediante un algoritmo de segmentación es una opción para paliar la dificultad de procesar, analizar o explorar series largas debido a la gran cantidad de datos que pueden contener. Trabajos previos propuestos en Oliver and Forbes [1997], Oliver et al. [1998] sugieren dividir la serie temporal usando puntos de cambio, por ejemplo máximos y mínimo locales de la serie, identificados previamente, y sustituyendo los segmentos con aproximaciones adecuadas. No obstante, este Trabajo Fin de Máster se centra en la aproximación linear por tramos (*Piecewise Linear Approximation, PLA*), la cuál nos permite reducir el número de puntos de la serie con la mínima pérdida de información, es decir, cometiendo el mínimo error posible [Keogh et al., 2004].

Como se ha explicado previamente, es necesaria una función de *clustering* para poder agrupar los segmentos resultantes. Con el fin de hacer el agrupamiento determinista, nos hemos basado en el algoritmo propuesto en Arthur and Vassilvitskii [2007], en el que se usa un *k-means* eligiendo los centroides de forma determinista, ya que se usa una función de distribución a partir las distancias entre unos y otros.

Por último, cabe destacar que no existe un algoritmo multiobjetivo con la misma filosofía que el que se propone en el presente Trabajo Fin de Máster. Sin embargo, la estructura base del algoritmo propuesto en esta investigación se debe al trabajo de Deb et al. [2002].

2.2. Segmentación basada en *Clustering* y Descomposición *Wavelet* Discreta

2.2.1. Introducción

Este método propuesto en Tseng et al. [2009] consiste en la utilización de una descomposición *wavelet* discreta para caracterizar los segmentos. Esta descomposición, junto con el número de pendientes de cada segmento, es utilizada para luego realizar el *clustering*. En las siguientes secciones se explica con más detalle el algoritmo.

2.2.2. Detalles del Enfoque Propuesto

Representación del Cromosoma

Existen dos métodos para la codificación en los algoritmos genéticos. Uno es la cadena de bits mientras que el otro consiste en una representación real. En este enfoque, se ha elegido la codificación real para la representación del cromosoma. Una serie temporal con n puntos puede ser representada como $T = \{d_1, d_2, ..., d_k, ..., d_n\}$, dónde d_k denota el k-ésimo punto en la serie T. Una segmentación resultante puede ser representada mediante un número de puntos de corte. Por lo que un cromosoma está compuesto por los puntos de corte resultantes de la segmentación.

En la Figura 2.1 cada individuo está representado como una cadena de número reales con el alfabeto $c_1, c_2, ..., c_h, ..., c_p$ dónde c_h representa el punto de corte entre los puntos de la serie d_{ch} y d_{ch+1}. Es obvio, que la siguiente restricción siempre debe satisfacerse: $c_1 < c_2 < ... < c_p$.

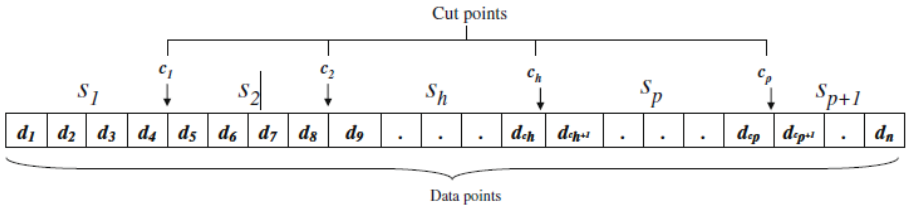


Figura 2.1: Representación del cromosoma para la Descomposición *Wavelet* Discreta

Por ello, el primer segmento irá desde d_1 a d_{c1}, el segundo segmento irá desde d_{c1+1} a d_{c2} y así sucesivamente, hasta llegar al último segmento compuesto por los puntos comprendidos entre d_{cp+1} a d_n.

Población Inicial

Todo algoritmo genético requiere una población inicial de soluciones factibles para posteriormente ir evolucionándola mediante las modificaciones realizadas por los operadores del algoritmo. Como se mencionó anteriormente, cada individuo dentro de una población es un posible resultado de la segmentación de una serie temporal determinada. Inicialmente, se generan un conjunto de cromosomas con algunas limitaciones para formar segmentos factibles. Se generan n puntos de corte de tal forma que cada segmento tenga la misma longitud. Posteriormente, se añaden ciertos valores aleatorios para mover hacia la izquierda o hacia la derecha los puntos de corte iniciales, y así formar los cromosomas iniciales.

Agrupando Segmentos en *clusters*

Con el algoritmo que estamos analizando, no sólo se pretende segmentar las series temporales si no también, agrupar los segmentos según su similitud. Existen métodos de *clustering* muy buenos ya propuestos ([Chen, 2005] y [Keogh et al., 2003]). El algoritmo *k-means* será utilizado para agrupar segmentos obtenidos a partir de un cro-

mosoma (serie temporal codificada) en k grupos. Por ello, es necesario encontrar las características adecuadas para la agrupación.

Supóngase el caso de que tenemos dos segmentos Q y S, cada uno de los cuáles contiene n puntos. Así pues, $Q = \{q_1, q_2, ..., q_n\}$ y $S = \{s_1, s_2, ..., s_n\}$ respectivamente. Para evaluar la similaridad de los dos segmentos , la distancia Euclídea de los puntos de los segmentos es un criterio que se utiliza a menudo:

$$ED(Q, S) = \frac{\sqrt{\sum_{k=1}^{n}(q_k - s_k)^2}}{n}, \tag{2.1}$$

dónde Q y S son los dos segmentos en un cromosoma, n es la longitud de los segmentos, q_k y s_k son los k-ésimos puntos en Q y S respectivamente. En el enfoque que se presenta, la longitud de los segmentos podría ser diferente, de forma que utilizar directamente los puntos no es posible. Por ello, se utilizarán las pendientes de las rectas de tendencia como atributos para agrupar segmentos. El usuario establece un parámetro $numSlopes$ que representa el número de pendientes para cada segmento. Se toman $(numSlopes + 1)$ pendientes de cada segmento para formar las pendientes indicadas por $numSlopes$. El primer y último punto son los extremos. El resto de puntos se seleccionan sistemáticamente de forma que entre dos puntos adyacentes exista la misma longitud. Luego se obtiene un valor de pendiente a partir de cada dos puntos adyacentes. Definimos las pendientes extraídas de un segmento S como Slp_1^s, Slp_2^s,..., $Spl_{numSlopes}^s$. El r-ésimo valor de pendiente (Slp_r^s) de S se calcula mediante los puntos r-ésimo y el r-ésimo más uno en S. Simplificando, el r-ésimo punto seleccionado en S será representado como $s_{r'}$. La pendiente de Slp_r^s se calcula con la siguiente fórmula:

$$Slp_r^S = (S_{(r+1)'} - S_{r'})/((r+1)^{'} - r^{'}). \tag{2.2}$$

Las pendientes son utilizadas para evaluar la distancia entre los dos

segmentos S y Q. La fórmula para la evaluación es la siguiente:

$$ED_{slopes}(Q,S) = \frac{\sqrt{\sum_{r=1}^{numSlopes}(Slp_r^Q - Slp_r^S)^2}}{numSlopes}, \qquad (2.3)$$

dónde Q y S son dos segmentos en un cromosoma y Slp_r^Q y Slp_r^S son los r-ésimos valores de pendientes de Q y S. ED_{slopes} es utilizada después como una característica para el *clustering*.

Descomposión *Wavelet* Discreta

Como se ha indicado anteriormente, los segmentos de un cromosoma pueden tener una longitud diferente. Esto hace que sea necesario diseñar una aproximación para ajustarlos de forma que todos tengan la misma longitud, para calcular sus similitudes. Para ello, se recurre a la «Transformación *Wavelet* Discreta» [Percival and Walden, 2000]. Dada una serie temporal de entrada, la transformada *wavelet* de Haar, que es la transformada más simple, se utiliza para emparejar los valores de entrada. Se define T como una serie temporal con n puntos, representada como $T = \{d_1, d_2, ..., d_k, ..., d_n\}$. Se genera un conjunto de coeficientes y una aproximación en un paso de la transformación. La aproximación es calculada como $(d_i + d_{i+1})/2$ y los coeficientes como $(d_i - d_{i+1})/2$. La aproximación resultante es utilizada en el siguiente paso de la transformada. Este proceso, por tanto, se repite recursivamente y se generan finalmente un total de $2^n - 1$ coeficientes y una aproximación. La Figura 2.2 resume este procedimiento.

Selección y Función de Aptitud (*fitness*)

Con el fin de conseguir un buen resultado de la segmentación de una población inicial dada, el algoritmo genético selecciona buenos «padres» en cuanto a segmentación para ser cruzados y mutados, y así obtener la siguiente generación. Para ello, tanto el método de la ruleta como otros métodos elitistas podrían ser utilizados como estrategias para conseguir una mejor siguiente generación. Obviamente, es

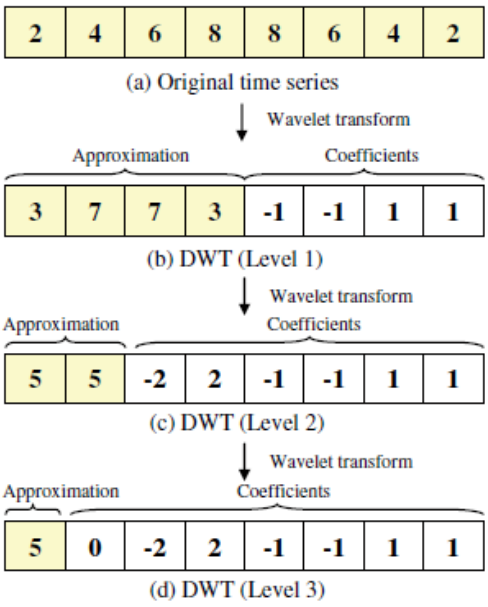


Figura 2.2: Ejemplo del proceso DWT

necesaria una función de evaluación de las segmentaciones resultantes que se van obteniendo de una generación a otra.

El valor de distancia $D(seg_{qj}^h)$ del segmento j en el *cluster* h del cromosoma C_q es definido como:

$$D(seg_{qj}^h) = Min_{g \neq j}(Dist(seg_{qj}^h, seg_{qg}^h)), \tag{2.4}$$

dónde $Dist(seg_{qj}^h, seg_{qg}^h)$ es la distancia euclídea de los dos segmentos seg_{qj}^h y seg_{qg}^h. Cuando estos segmentos tienen diferente longitud, la transformada *wavelet* discreta es aplicada al segmento de mayor longitud hasta conseguir la misma longitud que el segmento más corto. Estos segmentos resultantes son los utilizados para calcular la distancia entre ellos, en lugar de los originales.

Por tanto, $D(seg_{qj}^h)$ representa la mínima distancia entre seg_{qj}^h y cualquier otro segmento en el mismo *cluster*. El segmento con la mínima distancia a seg_{qj}^h , también puede ser considerado el más similar a él. Ahora se debe definir la distancia $ClusterD(Cluster_q^h)$ del *cluster*

h en el cromosoma C_q como:

$$ClusterD(Cluster_q^h) = \sum_{j=1}^{sn_q^h} D(seg_{qj}^h), \tag{2.5}$$

dónde sn_q^h es el número de segmento en el *cluster* h en el cromosoma C_q. El valor de $ClusterD$ de este modo representa la similaridad entre los segmentos de un *cluster*.

Operadores Genéticos

Los operadores genéticos son fundamentales en el algoritmo. En este algoritmo, se utiliza el operador de cruce en un punto y el operador de mutación en un punto [Chung et al., 2004].

El operador de cruce en un punto consiste en elegir un punto de corte al azar y dados dos individuos o padres (en este caso dos segmentaciones) combinar ambos padres a partir de ese punto, intercambiando las posiciones de la izquierda y derecha de dicho punto. Es decir, si tenemos dos cromosomas y un punto de corte, los cromosomas resultantes se obtendrán de la siguiente forma:

- El primer hijo estará formado por la parte izquierda al punto de corte del primer cromosoma y por la parte derecha al punto de corte del segundo cromosoma.
- El segundo hijo estará formado por la parte izquierda al punto de corte del segundo cromosoma y por la parte derecha al punto de corte del primer cromosoma.

El operador de mutación en un punto consiste en eliminar o añadir un punto de corte dado de forma aleatoria a un cromosoma.

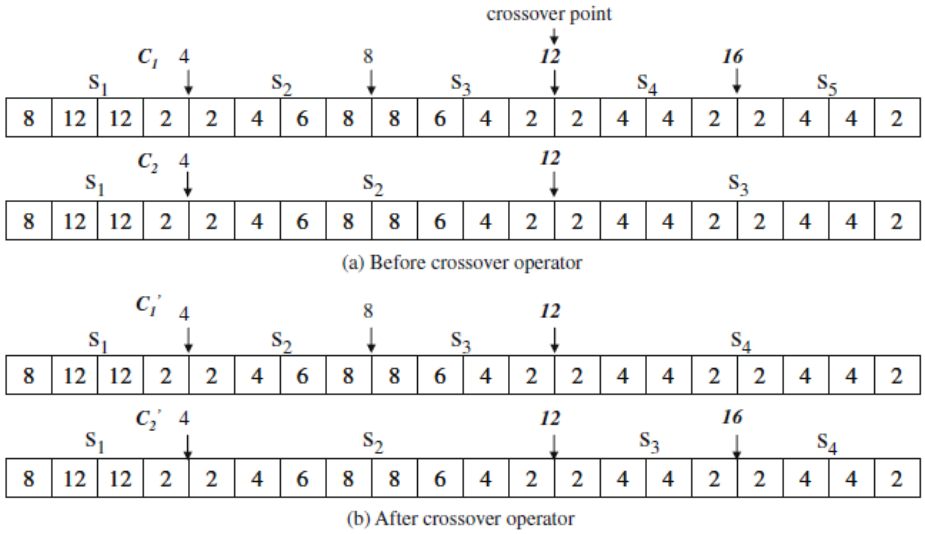


Figura 2.3: Operador de cruce para la Descomposición *Wavelet* Discreta

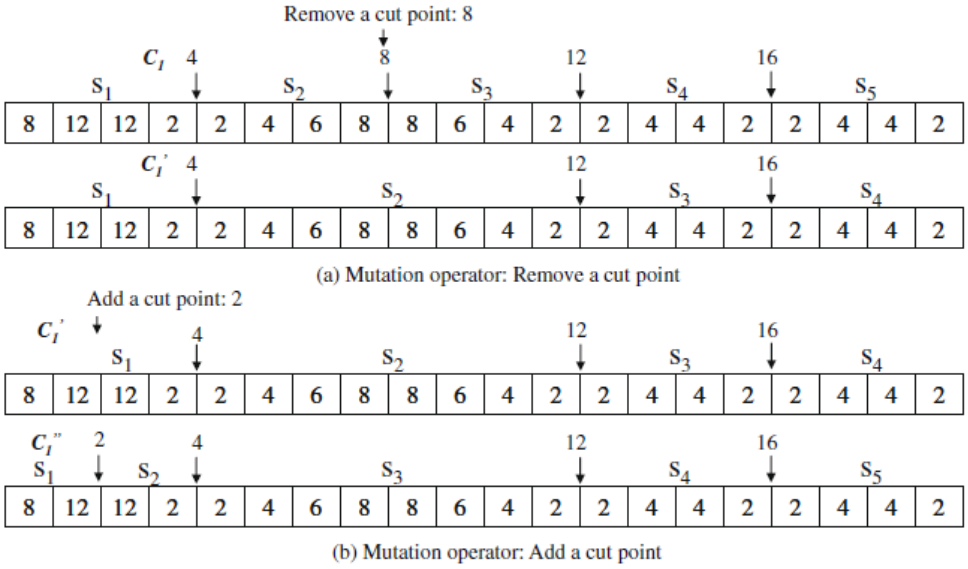


Figura 2.4: Operador de mutación para la Descomposición *Wavelet* Discreta

2.2.3. Pseudocódigo del Algoritmo

```
Algoritmo Seg_T = evol(T)
  Generación de la población inicial de tamaño PopSize.
    Cada individuo representa una segmentación factible
     de la serie T.
  Dividir los segmentos de cada cromosoma en k clusters:
    a) Calcular el número de pendientes de cada segmento.
       Cada segmento S es transformado a una numSlopes–
       tupla de pendientes (Slp_1^S, Slp_2^S, ..., Slp_numSlopes^S).
    b) Seleccionar aleatoriamente k segmentos como
       centros iniciales.
    c) Asignar cada segmento S a uno de los k grupos de
       acuerdo a la distancia euclídea de los valores de
       las pendientes.
    d) Recalcular los centroides como la media de los
       valores de los segmentos de cada grupo.
    e) Si los centroides no cambian pasar al siguiente
       paso, si cambian volver al paso c).
  Calcular el fitness de cada cromosoma mediante la fór–
    mula f(C_q) = sum_{h=1}^{k} ClusterD(Cluster_q^h)
  Aplicar operador de cruce a la población.
  Aplicar operador de mutación a la población.
  Seleccionar a los individuos destinados a reproducirse
    en la siguiente generación. Se usará el método de
    la ruleta o bien otro procedimiento elitista.
  Si se satisface la condición de parada se devuelve el
    mejor cromosoma, en caso contrario, se vuelve al
    paso 2.
```

2.3. Segmentación basada en *Clustering* Mono-objetivo

Ésta metodología es similar a la anterior. Fue propuesta en 2014 por el autor del presente trabajo en colaboración con investigadores del grupo de invesigación AYRNA y de la Agencia Europea Espacial [Nikolaou et al., 2014]. En este trabajo, en lugar de utilizar el número

de pendientes y la descomposición *wavelet* discreta para comparar los subsegmentos, se propuso la utilización de un conjunto de características de los segmentos para poder ser comparados.

2.3.1. Características del Algoritmo

Representación de los Individuos

La codificación elegida para representar cada una de las soluciones es directa. Es decir, la solución está formada por un vector de enteros, de forma que cada posición almacena cada uno de los cortes que se producen en la serie para dicha solución.

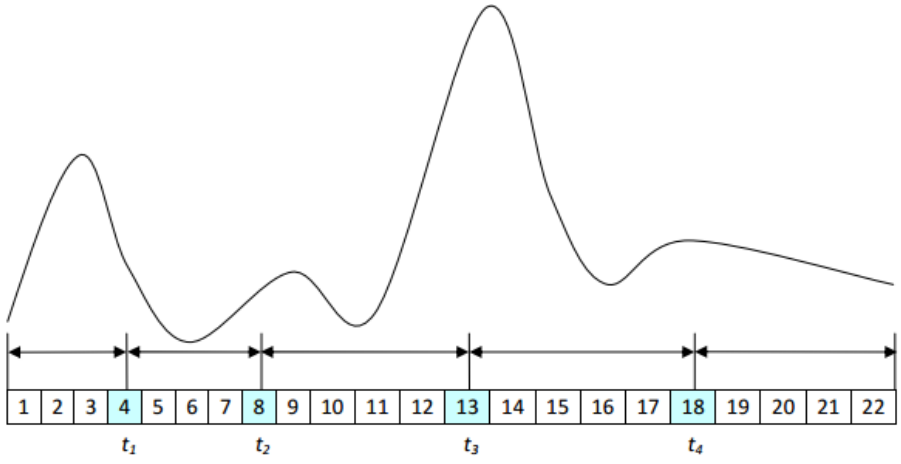


Figura 2.5: Representación del cromosoma del algoritmo mono-objetivo evolutivo

Generación de la Población Inicial

La población inicial se genera de forma aleatoria para un número predefinido t de individuos. Este número inicial de individuos se corresponde con el total de los individuos utilizados en el algoritmo evolutivo.

Función de *Fitness*

La función de *fitness* utilizada en el algoritmo evolutivo es la siguiente:

$$A = SSE/num_{segmentos}. \tag{2.6}$$

Se consideran una serie de características para representar cada uno de los segmentos de la serie, con independencia de su longitud. En estas características se basa el algoritmo *k-means* y son las siguientes:

Varianza Es una medida de variabilidad que da cuenta del grado de homogeneidad de un grupo de observaciones:

$$\sigma^2 = (\frac{1}{N} \sum_{i=limInf}^{limSup} {x_{t_i}}^2) - {\bar{x_t}}^2, \tag{2.7}$$

siendo N el número de puntos, *limInf* el punto inferior del comienzo del segmento, *limSup* el punto superior de final del segmento, x_{t_i} cada uno de los puntos del segmento, y $\bar{x_t}$ la media de los valores del segmento.

Coeficiente de Asimetría γ_1 Es una medida que indica si los valores del segmento presentan la misma forma a izquierda y derecha de la media aritmética:

$$\gamma_1 = \frac{\frac{1}{N} \sum_{i=limInf}^{limSup} (x_{t_i} - \bar{x_t})^3}{\sigma^3}, \tag{2.8}$$

siendo σ la desviación típica.

Coeficiente de Apuntamiento γ_2 Analiza el grado de concentración que presentan los valores alrededor de la zona central de la

distribución:

$$\gamma_2 = \frac{\frac{1}{N}\sum_{i=limInf}^{limSup}(x_{t_i} - \bar{x_t})^4}{\sigma^4} - 3. \tag{2.9}$$

Pendiente y Ordenada en el origen de una Recta de Regresión $y = ax + b$ Se calcula una recta de tendencia del segmento o recta de regresión de la forma $y = ax + b$. Para calcular el valor de a se aplica la siguiente fórmula:

$$a = \frac{S_{tx_t}}{{S_t}^2}, \tag{2.10}$$

siendo S_{tx_t} la covarianza entre los valores de cada instante t y el valor de la serie temporal en ese instante x_t; y S_t la desviación típica de los valores de tiempo. La expresión matemática de la covarianza sería la siguiente:

$$S_{tx_t} = \frac{1}{N}\sum_{i=1}^{N}(t_i - \bar{t}) \cdot (x_{t_i} - \bar{x_t}). \tag{2.11}$$

El parámetro b es calculado como:

$$b = \bar{y} - \hat{a}\bar{x}, \tag{2.12}$$

siendo $\hat{a}$ el valor de la pendiente de la recta calculada previamente.

Error Cuadrático Medio de Regresión *MSE* Una vez calculada la recta de regresión, se considera como característica del segmento el error cuadrático medio producido por dicha recta. Para ello, se ha empleado la siguiente expresión:

$$MSE = S_{x_t}^2 \cdot (1 - r^2), \tag{2.13}$$

dónde:

$$r^2 = \frac{S_{tx_t}}{{S_{x_t}}^2 \cdot {S_t}^2}. \tag{2.14}$$

Número de Pendientes Otra de las características consideradas para los segmentos es el número de pendientes existentes en el mismo. Es importante destacar que esta característica cuenta el número de pendientes en general que tiene el segmento sin considerar cuáles son las positivas o negativas.

Diferencia entre Valor Máximo y Mínimo Se trata de una característica que permite observar cuál es la amplitud del rango de valores de la serie en ese segmento. Se calcula de acuerdo a la siguiente fórmula:

$$DifMaxMin = Valor_{max} - Valor_{min}, \tag{2.15}$$

siendo $Valor_{max}$ y $Valor_{min}$ el punto máximo y el mínimo de la serie en el segmento considerado.

Coeficiente de Autocorrelación Permite conocer cuál es la relación entre un punto y el siguiente de la serie mediante la expresión:

$$CA = \frac{\sum_{i=1}^{N-1}(x_{t_i} - \bar{x_t}) \cdot (x_{t_i+1} - \bar{x_t})}{S_{x_t}^2}. \tag{2.16}$$

Longitud Está característica representa el rango de tiempo que abarca el segmento:

$$Longitud = t_f - t_i, \tag{2.17}$$

dónde t_f es el instante de tiempo final, y t_i, el instante de tiempo inicial.

Algoritmo de *Clustering k-means* Para la obtención del valor de la función de *fitness* es necesario el cálculo previo de la "bondad" del agrupamiento de los segmentos, de la segmentación de la serie realizada, utilizando uno de los algoritmos existentes. Se utiliza el clásico algoritmo *k-means*.

Selección y Reemplazo Se utiliza como técnica de selección, la selección directa tomando como criterio la «selección de todos los individuos». Es decir, en cada generación todos los individuos de la población serán destinados a reproducirse. De esta forma, se añade una mayor diversidad al algoritmo ya que no se tiene ningún criterio elitista para la selección de padres. En el proceso de reemplazo se hace uso de una selección por ruleta.

Operador de Mutación

El algoritmo utiliza un operador de mutación cuya función primordial es realizar una mejor exploración aleatoria del espacio de búsqueda. De esta forma, el algoritmo reduce su dependencia de la población inicial, a la vez que escapa de los posibles óptimos locales que se puedan alcanzar durante su ejecución. El operador de mutación implementado consiste en añadir, eliminar, desplazar hacia a la izquierda o desplazar hacia la derecha un número de puntos de corte que se le indican al mismo. La Figura 2.6 muestra este esquema de mutación.

Operador de Cruce

El algoritmo también utiliza un operador de cruce cuya función primordial es realizar una explotación de las soluciones existentes hasta el momento. El operador de cruce implementado es binario, es decir, sólo afecta a dos progenitores. El cruce consiste en determinar si un individuo se debe cruzar bajo una probabilidad de cruce que es especificada por el usuario, mientras que el segundo padre es elegido al azar de forma que no coincida con el primero. Un ejemplo se puede

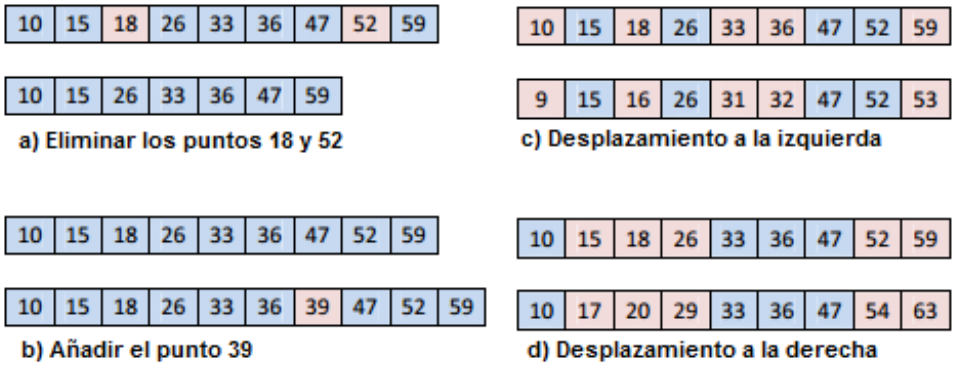


Figura 2.6: Operador de mutación del algoritmo mono-objetivo evolutivo

observar en la Figura 2.7.

Se determina un punto de corte, de modo que se intercambian entre los dos padres las posiciones del cromosoma (vector de enteros que representa la solución de los puntos de corte) superiores a dicho punto de corte.

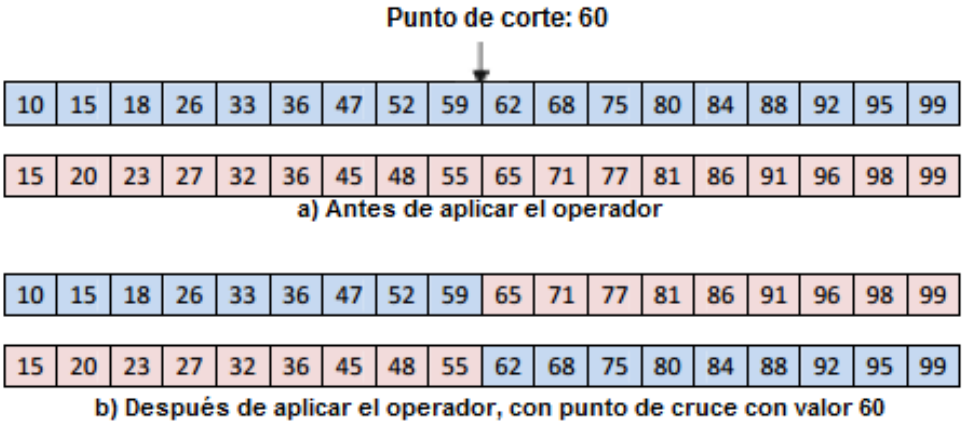


Figura 2.7: Operador de cruce del algoritmo mono-objetivo evolutivo

2.3.2. Pseudocódigo del Algoritmo

```
Algoritmo Seg_T = evolTSS(T)
  Genera una población aleatoria de tamaño N
  Evaluar todas las segmentaciones de la población inicial
      usando la función de fitness
  Mientras no se satisfaga la condición de parada
    Almacenar una copia de la mejor segmentación
    Seleccionar los padres para reproducirse
    Generar la población de hijos: Aplicar los operadores
        de cruce y mutación, en este orden
    Evaluar la población de hijos usando la función de
        fitness
      - Extracción de características
      - Proceso de clustering
      - Cálculo del la función de fitness SSE
    Proceso de reemplazamiento
  FinMientras
  Devolver la mejor segmentación
Fin
```

2.4. Segmentación basada en Aproximaciones PLA (*Piecewise Linear Approximation*) de las Series Temporales

El trabajo de Keogh et al. [2004] propone una serie de métodos para la segmentación de series temporales basándose en la bondad de la aproximación usando el subconjunto de puntos producidos por dicha segmentación. Todos los algoritmos necesitan como parámetro de entrada el error máximo que se puede cometer en cada uno de los segmentos. El error cometido por cada punto se puede calcular como la diferencia entre un punto de la serie original y su aproximación $e_i = t_i - \bar{t}_i$, siendo t_i el punto i de la serie original T, e $\bar{t}_i$ el de la aproximación. Así, el error de cada segmento puede ser cualquiera que

Notación	Significado
T	Serie temporal en la forma t_1,t_2, ..., t_n.
T[a:b]	Segmento de T entre a y b: t_a,t_{a+1}, ..., t_b.
Seg_TS	Aproximación PLA de la serie, cada segmento: Seg_TS(i).
crear_segmento(T)	Función que crea una aproximación de una sucesión de puntos.
calcular_error(T)	Devuelve el error de aproximación de una serie.

Tabla 2.1: Notación para los pseudocódigos.

se desee (suma de errores cuadráticos, valor medio, etc).

A continuación se muestran los cuatros algoritmos que propone dicho método y sus pseudocódigos. La Tabla 2.1 resume la notación que se va a utilizar.

2.4.1. Algoritmo *Sliding Window*

Este algoritmo consiste en considerar el primer punto como segmento e ir aumentando el tamaño, añadiendo sucesivamente los puntos de la serie, hasta que se sobrepase el máximo error permitido. Momento en el que, comenzará a repetirse el proceso hasta llegar al final de la serie temporal. El pseudocódigo es el siguiente:

```
    Algoritmo Seg_TS = Sliding_Window(T, max_error)
      ancho = 1;
      Mientras no acabe la segmentacion
        i=2;
 5      Mientras calcular_error(T[ancho:ancho+i]) < max_error
          i = i + 1;
        FinMientras
        Seg_TS = concatenar(Seg_TS, crear_segmento(T[ancho:
            ancho+(i-1)]);
        ancho = ancho + i;
10    FinMientras
    Fin
```

2.4.2. Algoritmo *Top-Down*

Este algoritmo comienza considerando toda la serie como un segmento. Después continua cortando la serie en el punto que cometa el mínimo error de aproximación. El algoritmo continua recursivamente para cada uno de los subsegmentos generados, hasta que el segmento tiene un error más pequeño que el máximo permitido. El pseudocódigo es el siguiente:

```
    Algoritmo Seg_TS = Top_Down(T, max_error)
      mejora_actual = inf;
      Para i = 2 hasta que i = longitud(T) - 2
4       mejora_aproximacion = mejora_si_corto(T,i)
        Si mejora_aproximacion < mejora_actual
          punto_corte = i;
          mejora_actual = mejora_aproximacion;
        FinSi
9     FinPara

      Si calcular_error(T[1:punto_corte]) > max_error
        Seg_TS = Top_Down(T[1:punto_corte]);
      FinSi
14
      Si calcular_error(T[punto_corte+1:longitud(T)])>
         max_error
        Seg_TS = Top_Down(T[punto_corte+1:longitud(T)]);
      FinSi
    Fin
```

2.4.3. Algoritmo *Bottom-Up*

Es el algoritmo complementario por naturaleza al *Top-Down*. Comienza considerando cada par de puntos como un segmento. En las sucesivas iteraciones, se van uniendo aquellos segmentos que, tras dicha unión, cometan el mínimo error posible. El pseudocódigo es el siguiente:

```
    Algoritmo Seg_TS = Bottom_Up(T, max_error)
 2    Para i = 1 hasta que i = longitud(T), con un paso de i
        = i + 2
      Seg_TS = concat(Seg_TS, crear_segmento(T[i:i+1]));
      FinPara
      Para i = 1 hasta que i = longitud(Seg_TS) - 1
        coste_union(i) = calcular_error(unir(Seg_TS(i),Seg_TS
          (i+1)));
 7    FinPara

      Mientras minimo(coste_union) < max_error
        indice = minimo(coste_union);
        Seg_TS(indice) = unir(Seg_TS(indice), Seg_TS(indice
          +1));
12      eliminar(Seg_TS(indice+1));
        coste_union(indice)=calcular_error(unir(Seg_TS(indice
          ), Seg_TS(indice+1)));
        coste_union(indice-1)=calcular_error(unir(Seg_TS(
          indice-1), Seg_TS(indice)));
      FinMientras
    Fin
```

2.4.4. Algoritmo *Sliding Windows and Bottom-Up* (SWAB)

El **SWAB** se trata de un algoritmo que combina los algoritmos *Sliding Windows* y *Bottom-Up*. Para este algoritmo, es necesario un parámetro auxiliar que es el tamaño de la ventana en el instante inicial. Así el algoritmo comienza realizando un *Bottom-Up* en los puntos de esa ventana, toma el primer segmento que se forma dentro, y el siguiente punto a éste lo considera como primer punto de la ventana para la iteración siguiente. En la misma iteración, se aplica una ventana deslizante para los puntos que no pertenecen a la ventana, y el primer segmento generado marcará el límite del fin de la ventana de la siguiente iteración. El proceso se repite hasta que se recorren todos los puntos. El pseudocódigo es el siguiente:

```
    Algoritmo Seg_TS = SWAB(max_error, tam_buffer)
      izq = 1;
      der = tam_buffer;
4     Mientras no se recorran todos los puntos
        Seg_TS[izq:der] = Bottom_Up(T[izq:der]);
        Seg_TS[der+1:longitud(T)] = Sliding_Window(T[der+1:
            longitud(T)]);
        izq = limite_derecho(primer_segmento(Bottom_Up(T[izq:
            der])));
        der = limite_derecho(primer_segmento(Sliding_Window(T
            [der+1:longitud(T)])));
9     FinMientras
    Fin
```

La Figura 2.8 muestra el diagrama de flujo del algoritmo SWAB.

2.5. Algoritmo *Nondominated Sorting Genetic Algorithm 2* (NSGA2)

En Deb et al. [2002], se presentan los detalles del *Nondominated Sorting Genetic Algorithm 2* (NSGA2), un algoritmo basado en clasificación por no dominancia para asignar el valor de adaptabilidad a los elementos de la población genética. Son varias las características que lo hacen diferente del NSGA original. NSGA2 incorpora un mecanismo de preservación de elites que asegura el mantenimiento de las buenas soluciones encontradas con anterioridad. Además utiliza un procedimiento rápido de clasificación por no dominancia (*fast nondominated sorting procedure*), el cual incorpora un procedimiento especial de almacenamiento a fin de reducir la complejidad computacional. Por último, NSGA2 no requiere de ningún parámetro ajustable.

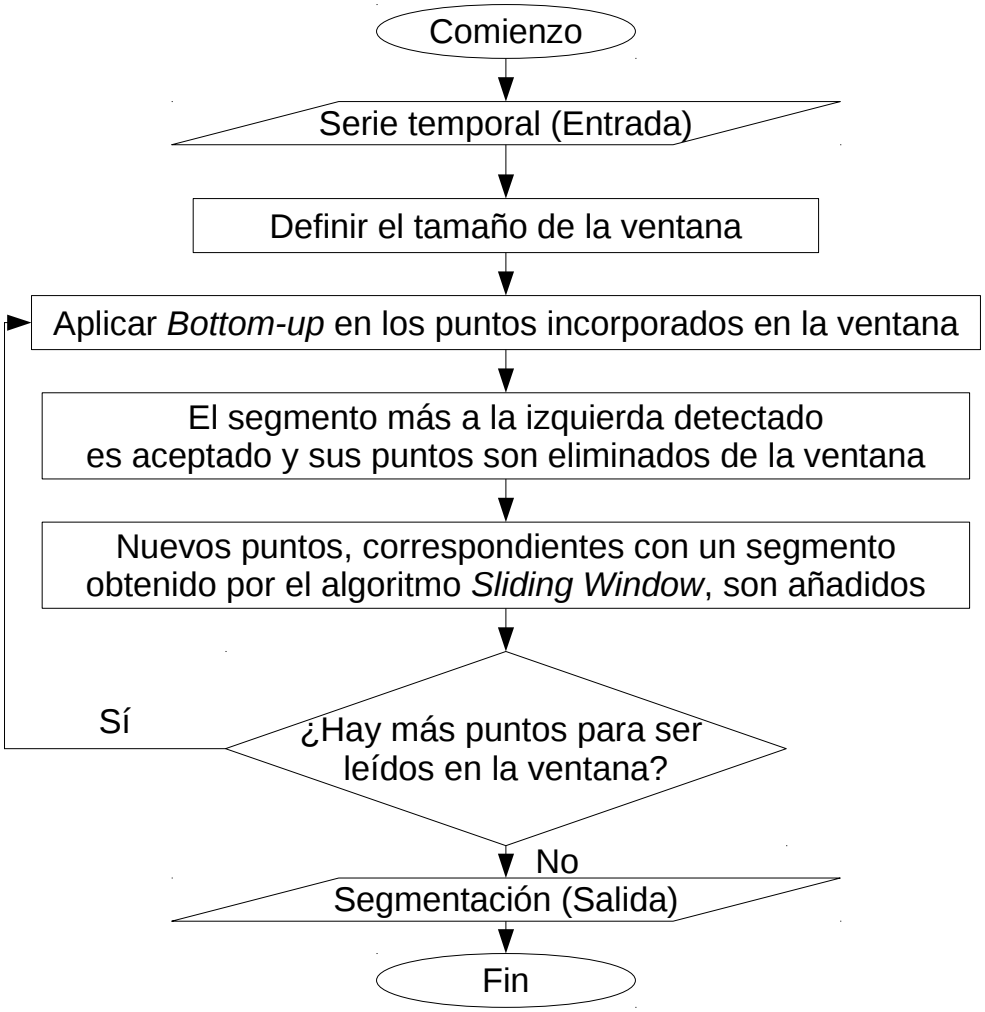


Figura 2.8: Diagrama de flujo del algoritmo SWAB

2.5.1. Conceptos Previos en la Optimización Multiobjetivo

A continuación, se definen una serie de conceptos esenciales en la optimización multiobjetivo.

Definición 1. Problema de optimización multiobjetivo (MOP): El problema de optimización multiobjetivo se puede formular, de forma general, como el problema de encontrar el vector $\mathbf{x}^* = [x_1^*, x_2^*, \cdots, x_n^*]^T$

que satisfaga las m restricciones de desigualdad:

$$g_i(\mathbf{x}) \geq 0 \quad i = 1, 2, \cdots, m \tag{2.18}$$

Las p restricciones de igualdad

$$h_i(\mathbf{x}) = 0 \quad i = 1, 2, \cdots, p$$

y que optimice

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_k(\mathbf{x})]^T$$

La solución de un MOP minimiza o maximiza (según corresponda) los componentes del vector $\mathbf{f}(\mathbf{x})$. Por tanto, el problema tiene k objetivos y las funciones $\mathbf{f}(\cdot) : \Omega \rightarrow A$ representan la correspondencia entre el espacio de búsqueda Ω y el espacio de las funciones objetivo A. De la definición de MOP se infiere que la solución del problema puede no ser única, ya que los diversos objetivos pueden estar en conflicto, de forma que la optimización de uno de ellos lleva, en general, a un descenso del valor de otro.

Definición 2. Optimalidad de Pareto: Decimos que un punto $\mathbf{x}^* \in \Omega$ es un óptimo de Pareto si para todo $\mathbf{x} \in \Omega \quad e \quad I = \{1, 2, \cdots, k\}$ se verifica que $\forall_i \in I, (f_i(\mathbf{x})) = f_i(\vec{x}^*))$ o $\exists\ i \in I$ tal que $f_i(\mathbf{x}) > f_i(\mathbf{x}^*)$.

Definición 3. Dominancia de Pareto: En un problema de minimización, un vector $\mathbf{u} = (u_1, u_2, \cdots, u_k) \in A \subseteq \Re^n$ domina a otro $\mathbf{v} = (v_1, v_2, \cdots, v_k) \in A \subseteq \Re^n$ (denotado mediante $\mathbf{u} \succeq \mathbf{v}$), si y solo si, u es parcialmente menor a v, es decir, $\forall_i \in \{1, \cdots, k\}, \quad u_i \leq v_i \wedge \exists \quad i \in \{1, \cdots, k\}$ tal que $u_i < v_i$.

Definición 4. Conjuntos de óptimos de Pareto: Para un problema multiobjetivo dado $\mathbf{f}(x)$, el conjunto de óptimos de Pareto ($\mathcal{P}^*$)

se define como:

$$\mathcal{P}^* := \{x \in \Omega \mid \nexists x' \in \Omega \quad \text{tal que } \mathbf{f}(x) \preceq \mathbf{f}(x')\}$$

Definición 5. Frente óptimo de Pareto: Para un problema multiobjetivo dado $\mathbf{f}(x)$ y un conjunto de óptimos de Pareto $\mathcal{P}^*$, el frente de Pareto ($\mathcal{PF}^*$) se define como:

$$(\mathcal{PF}^*) := \{\mathbf{u} = \mathbf{f} = (f_1(x), \cdots, f_k(x)) \,| x \in \mathcal{P}^*\}$$

Definición 6. Mínimo global de un MOP: Dado un vector de funciones $\mathbf{f}(\cdot) : \Omega \subseteq \Re^k \rightarrow \Re^n$, $\Omega \neq 0$, y $k \geq 2$, se llama al conjunto $\mathcal{PF}^* : \mathbf{f}(\mathbf{x}^*)$ mínimo global, si y sólo si, $\forall \mathbf{x} \in \Omega : \mathbf{f}(\mathbf{x}^*) \prec \mathbf{f}(\mathbf{x})$, donde $\mathbf{x}^* \in \Omega$ se llama conjunto de soluciones de mínimo global, $\mathbf{f}(\cdot)$ es el vector de funciones objetivos y Ω el espacio de búsqueda.

A raíz de estas definiciones se deduce que el mínimo global es la frontera de Pareto del problema y las soluciones de mínimo global forman el conjunto de óptimos de Pareto. Un problema de optimización de Pareto donde el espacio de búsqueda es $\Re^k$, tiene, en general, infinitas soluciones.

Definición 7. Dominancia débil y fuerte: Un vector es un óptimo de Pareto débil si no existe otro vector para el cual todos sus componentes en el espacio de las funciones objetivo sean mejores. De manera formal, se puede definir como sigue:

- **No dominancia débil**: Una solución $\mathbf{x}^* \in \Omega$ es una solución débilmente no dominada si no existe otra solución $\mathbf{x} \in \Omega$ tal que $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$, para $i = 1, 2, \cdots, k$.
- **No dominancia fuerte**: Una solución $\mathbf{x}^* \in \Omega$ es una solución fuertemente no dominada si no existe otra solución $\mathbf{x} \in \Omega$ tal que $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$, para $i = 1, 2, \cdots, k$ y existe al menos un valor j para el cual $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$.

Así, por ejemplo, si consideramos un problema de minimización de una función de dos dimensiones $\mathbf{f}(\mathbf{x}) \in A \subseteq \Re^2$, los cuatro puntos dibujados y señalados en la figura 2.9 como Frente de Pareto, son cuatro posibles soluciones al problema. A la hora de elegir cuál de ellos es mejor, solamente podemos decir que cualquiera de ellos es mejor que los puntos que están fuera del frente (puntos rosas), pero que ninguno de ellos es mejor que otro del frente. En este ejemplo se produce una minimización, de forma que cualquier punto exterior al frente tiene al menos un valor mayor en uno de los objetivos, y un valor mayor o igual en el otro, comparado con el valor de los objetivos de los puntos del frente, por lo que son peores soluciones.

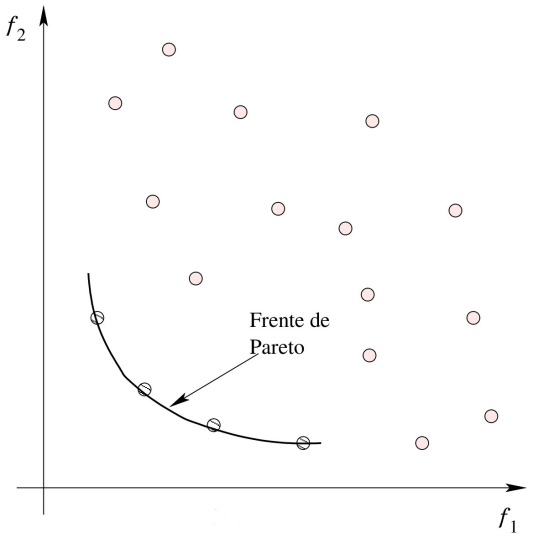


Figura 2.9: Ejemplo de frente de Pareto con dos objetivos a minimizar.

2.5.2. Ordenación Rápida de No Dominados

En el pseudocódigo de esta sección, se muestra el procedimiento rápido de ordenación por no dominancia sobre el cual basa su funcionamiento NSGA2. En este algoritmo, en primer lugar, se determina para cada solución $P[i]$ de una población P a clasificar:

1. El conjunto S_i de las soluciones dominadas por $P[i]$.

2. El número nd_i de soluciones que dominan a $P[i]$.

Para ello, se comparan entre sí con respecto a la dominancia todos los miembros de la población. Si un elemento $P[i]$ domina a un elemento $P[j]$, este último se agrega a un conjunto S_i. En caso contrario, si $P[i]$ es dominado por $P[j]$ entonces se incrementa el valor del contador nd_i de soluciones que dominan a $P[i]$. Notemos que un elemento $P[j]$ de la población puede pertenecer a nd_j conjuntos de soluciones dominadas.

Una vez determinados tanto el conjunto de soluciones dominadas como el número de soluciones que lo dominan, para cada elemento de la población, se forma el primer frente de soluciones no dominadas, denotado por $\mathbf{F^1}$, con todos los elementos cuyo contador de individuos que lo dominan es igual a 0. El algoritmo prosigue recorriendo para cada elemento $P[i] \epsilon \mathbf{F^1}$ su respectivo conjunto de soluciones dominadas S_i, reduciendo para cada elemento $P[j] \epsilon S_i$ el valor de nd_j que le corresponde.

Cuando el valor de nd_j se hace igual a 0, se agrega $P[j]$ a una lista $\mathbf{H}$ inicialmente vacía. Cuando se han recorrido todos los elementos del primer frente, en $\mathbf{H}$ quedan los elementos que sólo son dominados por los elementos del primer frente, es decir, los elementos del segundo frente $\mathbf{F^2}$. Luego se consideran los elementos de $\mathbf{F^2}$ repitiendo el procedimiento para cada elemento de dicho frente. El procedimiento finaliza cuando no quedan elementos cuya cuenta de elementos que los domina se haga cero, esto es cuando todos los frentes han sido identificados y $\mathbf{H} = \phi$.

```
    Algoritmo F = fast_nondominated_sort(P)
      Para cada P[i] ϵ P
        Para cada P[j] ϵ P
          Si P[i] ≻ P[j]
5           S_i = S_i ∪ P[j]
          Sino Si P[j] ≻ P[i]
            nd_i = nd_i + 1
          FinSi
        FinPara
10      Si nd_i = 0
          F^1 = F^1 ∪ {P[i]}
        FinSi
      FinPara
      f = 1
15    Mientras F^f ≠ 0
        H = Φ
        Para cada solución P[i] ϵ F^f
          Para cada solución P[j] ϵ S_i
            nd_j = nd_j − 1
20          Si nd_j = 0
              H = H ∪ {P[j]}
            FinSi
          FinPara
        FinPara
25      f = f + 1
        F^f = H
      FinMientras
    Fin
```

2.5.3. Operador de Agrupamiento *crowding* para la Diversidad

NSGA2 incluye también un procedimiento para estimar la densidad de soluciones alrededor de cada solución particular $F^f[i]$ con respecto a los demás elementos del frente $\mathbf{F^f}$. El siguiente pseudocódigo ilustra el procedimiento.

```
    Algoritmo F^f = crowding_distance_assignment(F^f)
 2    l = ||F^f||_c
      Para cada i, F^f[i]distance = 0, FinPara
        Para cada objetivo j del total k considerado
          Ordenar F^f de acuerdo al objetivo j
          F^f[1]distance = F^f[l]distance = ∞
 7        Para i desde 2 hasta (l − 1)
            F^f[i]distance = F^f[i]distance+
                  +(F^f[i + 1]objetivo[j] − F^f[i − 1]objetivo[j])
          FinPara
        FinPara
12    FinPara
    Fin
```

Para cada individuo $F^i[i]$, se calcula un valor denotado por la expresión $F^f[i]distance$ que sirve como un estimador del tamaño del cuboide más grande que encierra la solución sin incluir ningún otro punto de la población (a esto se le llama distancia *crowding*). Se determina para cada individuo del frente considerado y para cada objetivo, calculando para cada uno de los individuos cuál es el siguiente elemento menor y el siguiente mayor dentro del frente (con respecto al objetivo "i"). Para ello se ordenan de menor a mayor los elementos del frente para cada uno de los objetivos. Luego, el valor de la distancia *crowding* de un elemento $F^f[i]$ se calcula sumando las distancias entre los individuos inmediatamente mayor y menor considerando cada objetivo. Notemos que los objetivos usualmente se encuentran expresados en unidades diferentes por lo que para obtener una estimación correcta es conveniente la normalización de los diferentes objetivos.

Además de definir un procedimiento de asignación *crowding*, se define también un operador de comparación por crowding representado mediante $\geq n$. El objetivo de este operador es guiar el proceso de selección en las diferentes etapas del algoritmo hacia un Frente de Pareto óptimo uniformemente distribuido.

Asumiendo que cada uno de los individuos en la población tiene dos atributos: la posición en la clasificación por no dominancia ($P[i]_{rank}$)

y su distancia local de *crowding* ($P[i]_{distance}$), se define el orden parcial $\geq n$ como:

$$P[i] \geq P[j] \; si \; (P[i]_{rank} < P[j]_{rank}) \; o \\ ((P[i]_{rank} = P[j]_{rank}) \; y \; (P[i]_{distance} > P[j]_{distance})) \tag{2.19}$$

donde $P[i]_{rank} = f$ si $P[i] \; \epsilon \; \mathbf{F^f}$.

Esto es, se define un orden lexicográfico con dos objetivos, con la posición en la ordenación por no dominancia como el de mayor importancia. Entonces, entre dos soluciones con diferente posición en la clasificación por no dominancia, se prefiere aquella con la clasificación más baja. De otra forma, si ambas soluciones están localizadas en el mismo frente, se prefiere la que está ubicada en una región con un menor número de puntos.

2.5.4. Pseudocódigo del Algoritmo

El pseudocódigo del algoritmo es el siguiente:

```
Algoritmo F = NSGA2()
  Generar una población P(t) de tamaño N en forma
      aleatoria
  Evaluar la población
  Aplicar el algoritmo de ordenación por no dominancia y
      el algoritmo de crowding
  Utilizar torneo binario para seleccionar elementos de
      P(t)
    - Se establecen N pares aleatorios de elementos de la
        población:
    - Se aplica el operador de comparación crowding en cada
        par
    - Se seleccionan los N vencedores del torneo
  Efectuar cruce y mutación y así generar Q(t)
  Mientras no se satisfaga la condición de parada
    f = 1
    R(t) = P(t) ∪ Q(t)
    F = fast_nondominated_sort(R(t))
    Mientras ||(P(t+1)||_c < N
      crowding_distance_assignment(F^f)
      P(t+1) = P(t+1) ∪ F^f
      f = f + 1
    FinMientras
  Ordenar de forma descendiente de acuerdo al operador ≥n
  Tomar los primeros N elementos de P(t+1)
  Seleccionar individuos de P(t+1) con torneo binario
  Aplicar cruce y mutación para obtener Q(t+1)
  t = t + 1
  FinMientras
Fin
```

3 Algoritmo propuesto

En este capítulo, se describirá de forma concisa el algoritmo desarrollado en este Trabajo Fin de Máster.

3.1. Introducción al Algoritmo

Dada una serie temporal $Y = \{y_n\}_{n=1}^{N}$, nuestro objetivo es dividir los valores de y_n en m segmentos consecutivos. Estos segmentos deben presentar comportamientos parecidos, además de aproximar adecuadamente la serie temporal. Para ello, los instantes de tiempo ($n = 1, \ldots, N$) se dividen en segmentos: $s_1 = \{y_1, \ldots, y_{t_1}\}, s_2 = \{y_{t_1}, \ldots, y_{t_2}\}, \ldots, s_m = \{y_{t_{m-1}}, \ldots, y_N\}$, donde los t_s son los diferentes puntos de corte indicados de forma ascendente ($t_1 < t_2 < \ldots < t_{m-1}$). Los puntos de corte son los únicos que pertenecen a dos segmentos distintos, el anterior y el posterior. El número de segmentos m y los valores de $t_i, i = 1, \ldots, m-1$, deben ser determinados por el algoritmo multiobjetivo evolutivo.

Después, la metodología intenta agrupar los segmentos en k *clusters*, donde $k < m$ es un parámetro definido por el usuario. Cada segmento será asociado a una etiqueta de clase, con k posibles etiquetas, $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$. Por otro lado, la metodología intenta aproximar los

puntos de la serie mediante regresiones polinomiales en cada uno de los m segmentos, tratando de minimizar el error.

3.2. Representación de los Individuos

En este Trabajo, se ha considerado cambiar la codificación de cada una de las soluciones, que se presentó en la seccion 2.3.1, a una codificación binaria. Es decir, cada solución c consiste en un vector de tamaño N dónde cada posición $(c_i, \ldots, c_N)$ almacena un 1 si se trata de un punto de corte, o un 0 en caso contrario. Por tanto, cada posición c_i almacena si el instante de tiempo t_i de la serie temporal representa un punto de corte de la solución. En este sentido, dado un segmento s_i delimitado por los puntos de corte t_s y t_{s+1} $(t_s < t_{s+1})$, los valores del cromosoma serán $c_s = 1$, $c_{s+1} = 1$ y $c_l = 0, \forall l | t_s < l < t_{s+1}$. Un ejemplo puede verse en la Figura 3.1.

Debido al desarrollo del algoritmo, que posteriormente se expondrá, dicha forma de representación agiliza el procedimiento, ya que los operadores de mutación y cruce podrán trabajar con ellas directamente sin hacer uso de estructuras auxiliares que provoquen una ralentización, no debida al algoritmo en sí, sino a la forma de representación. Además, cómo el lenguaje de programación utilizado es MATLAB, el hecho de poseer cromosomas del mismo tamaño hace que se puedan expresar operaciones de forma matricial, lo que reduce aún más el tiempo de cómputo.

3.3. Generación de la Población Inicial

La población inicial se genera aleatoriamente para un número N de elementos, predefinido de antemano. Como novedad, en la inicialización es necesario un tamaño mínimo de segmento (min_{size}) y un tamaño máximo (max_{size}). De forma, que al iniciar aleatoriamente los puntos de corte, éstos están separados por un valor en el intervalo

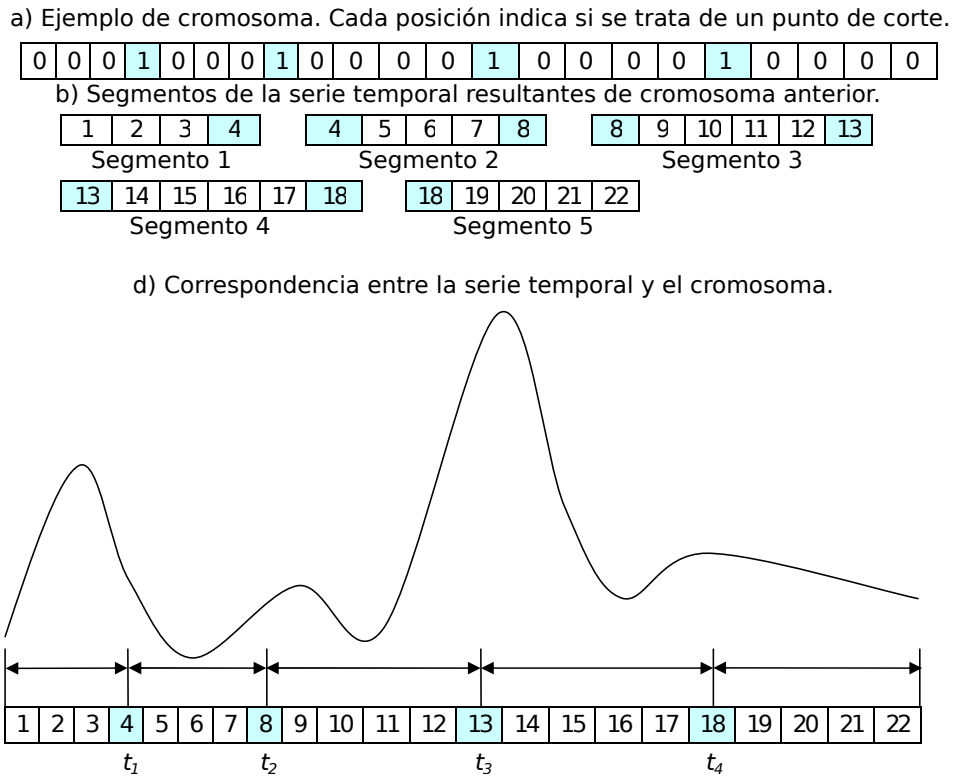


Figura 3.1: Representación del cromosoma del algoritmo multiobjetivo evolutivo

$[min_{size}, max_{size}]$.

3.4. Función de *Fitness*

Como se ha explicado en capítulos previos, se han desarrollado dos funciones de *fitness* contrapuestas. La primera, correspondiente a la función de *fitness* para el descubrimiento de patrones similares a lo largo de la serie; y la segunda, correspondiente a la minimización del error en la aproximación de la serie.

3.4.1. Función de *Fitness* para el Descubrimiento de Patrones Similares

Esta función de *fitness*, para el algoritmo multiobjetivo consiste en tres fases: extracción de características, aplicación de un algoritmo de *clustering* y medición de la bondad de dicho *clustering*.

Extracción de características

Dado que los segmentos en un cromosoma pueden tener distinto tamaño, una idea es la proyección de todos los segmentos en el mismo espacio dimensional. Para ello, en este trabajo se han considerado 4 métricas estadísticas fijas, y una serie de coeficientes dependiendo del grado del polimonio que se pretenda ajustar para obtener una función de regresión de los puntos del segmento. Así, si se considera un segmento s_s delimitado por los puntos de corte t_{s-1} y t_s, y con tamaño $t_s - t_{s-1} + 1$, el mapeo al mismo espacio dimensional se realiza mediante la función $f : \mathbb{R}^{(t_s - t_{s-1}+1)} \rightarrow \mathbb{R}^{4+n}$ (n es el grado del polinomio utilizado), de la siguiente forma:

$$f(s_s) = \left(S_s^2, \gamma_{1s}, \gamma_{2s}, AC_s, \beta_n, \beta_{n-1}, \ldots, \beta_1\right) \tag{3.1}$$

donde las diferentes características son definidas como:

1. Varianza (S_s^2): es una medida de variabilidad que da cuenta del grado de homogeneidad de un grupo de observaciones:

$$S_s^2 = \frac{1}{t_s - t_{s-1} + 1} \sum_{i=t_{s-1}}^{t_s} \left(y_i - \overline{y_s}\right)^2, \tag{3.2}$$

   donde y_i son los valores del segmento de la serie temporal, e $\overline{y_s}$ es la media de los valores del segmento.

2. Coeficiente de Asimetría (γ_{1s}): es una medida que indica si los valores del segmento presentan la misma forma a izquierda y

derecha de la media aritmética:

$$\gamma_{1s} = \frac{\frac{1}{t_s - t_{s-1}+1} \sum_{i=t_{s-1}}^{t_s} (y_i - \overline{y_s})^3}{S_s^3}, \tag{3.3}$$

siendo S_s la desviación estándar del s-éximo segmento.

- Si $\gamma_1 < 0$: La curva es asimétricamente negativa por lo que los valores se tienden a reunir más en la parte derecha de la media.
- Si $\gamma_1 = 0$: Se acepta que la distribución es simétrica, es decir, existe aproximadamente la misma cantidad de valores a los dos lados de la media. Este valor es difícil de conseguir por lo que se tiende a tomar los valores que son cercanos ya sean positivos o negativos ($\pm$0,5).
- Si $\gamma_1 > 0$: La curva es asimétricamente positiva por lo que los valores se tienden a reunir más en la parte izquierda que en la derecha de la media.

3. Coeficiente de Apuntamiento (γ_{2s}): analiza el grado de concentración que presentan los valores alrededor de la zona central de la distribución:

$$\gamma_{2s} = \frac{\frac{1}{t_s - t_{s-1}+1} \sum_{i=t_{s-1}}^{t_s} (y_i - \overline{y_s})^4}{S_s^4} - 3. \tag{3.4}$$

- Si $\gamma_2 = 0$: Se trata de una distribución mesocúrtica, que presenta un grado de concentración medio alrededor de los valores centrales de la variable.
- Si $\gamma_2 > 0$: Se trata de una distribución leptocúrtica, que presenta un elevado grado de concentración alrededor de los valores centrales de la variable.
- Si $\gamma_2 < 0$: Se trata de una distribución platicúrtica, que presenta un reducido grado de concentración alrededor de los valores centrales de la variable.

4. Coeficiente de autocorrelación (AC_s): permite conocer cuál es la relación entre un punto y el siguiente de la serie mediante la expresión:

$$AC_s = \frac{\sum_{i=t_{s-1}}^{t_s}(y_i - \overline{y_s}) \cdot (y_{i+1} - \overline{y_s})}{S_s^2}. \tag{3.5}$$

5. Coeficientes del polinomio de la regresión: se consideran como características, los coeficientes del polinomio resultante de obtener la función de regresión que mejor se adapta al conjunto de puntos de cada segmento mediante el método de los Mínimos Cuadrados. Así, según el grado del polinomio n que pretendamos ajustar, tendremos n coeficientes/características en nuestro mapeo, ya que, el término independiente no se incluye. En este caso, tendríamos la función de regresión similar a la del apartado 2.3.1 pero con un número mayor de coeficientes:

$$y = \beta_n x^n + \beta_{n-1} x^{n-1} + \cdots + \beta_1 x^1 + \beta_0, \tag{3.6}$$

siendo β's los coeficientes de la regresión, y n el grado del polinomio. El término independiente β_0 no se incluye como característica.

Algoritmo de *Clustering k-means*

Un algoritmo de *clustering* es aplicado para agrupar segmentos similares (teniendo encuenta las caracteríticas seleccionadas). Por simplicidad, el algoritmo elegido es el clásico y bien conocido *k-means*. Cabe destacar, que antes de aplicar el algoritmo de *clustering* es necesaria una normalización de las características para que todas se encuentren en el mismo rango de valores y por consecuente, tenga la misma influencia durante la ejecución del algoritmo de agrupamiento. Por ejemplo, la varianza puede tener un rango muchísimo más amplio de variación que el coeficiente de asimetría.

En el clásico *k-means*, los centroides iniciales son iniciados aleato-

riamente entre el conjunto de patrones. En su lugar, se ha desarrollado un proceso determinista para la selección de estos centroides iniciales, de forma que se asegure que un cromosoma siempre presentará el mismo valor de *fitness*. En primer lugar, se selecciona la característica que posea la máxima desviación estándar. El primer centroide será aquél segmento que presente el máximo valor en esta característica. El segundo será el segmento con la mayor distancia Euclídea con respecto el primero. El tercero el más alejado de los dos anteriores, y así sucesivamente. Este procedimiento garantiza una inicialización determinista además, de que los centroides están lo más separados posibles unos de otros, lo que garantiza la diversidad de los centroides.

Medición de la bondad del *clustering*

El último paso de la obtención de la función del *fitness* del agrupamiento, es medir cómo de bien se han agrupado los segmentos. Está claro que diferentes algoritmos de *clustering* obtienen diferentes agrupaciones. En este sentido, el problema de evaluar objetivamente y cuantitativamente los resultados es muy importante. Para ello, existen dos criterios diferentes para este propósito [Xu and Wunsch, 2008], el externo y el interno. El interno, se debe a evaluar los resultados de un *cluster* observando los datos agrupados en sí; mientras que la evaluación externa se refiere a usar, por ejemplo, etiquetas de clase. Basándonos en esos conceptos, la evaluación interna es más adecuada en nuestro problema ya que no se posee información a priori de los segmentos encontrados. Cabe destacar que las métricas de los segmentos han de estar normalizadas también en este proceso. Se han implementado un total de 10 índices:

- *Suma de errores cuadráticos (SSE)*: La métrica de error más sencilla es la suma de los errores cuadráticos (considerando errores a las distancia desde cada punto a su centroide):

$$SSE = \frac{1}{N} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{C}_i} d(\mathbf{x}, \mathbf{c}_i)^2, \tag{3.7}$$

donde k es el número de *clusters*, $\mathbf{c}_i$ es el centroide del *cluster* $\mathcal{C}_i$ y $d(\mathbf{x}, \mathbf{c}_i)$ es la distancia Euclídea entre el patrón $\mathbf{x}$ y el centroide $\mathbf{c}_i$. Esta función no tiene en cuenta que los *clusters* se formen muy cercanos unos de otros. Como este índice tiene que ser miminizado, la función de *fitness* se define como $f = \frac{1}{1+SSE}$.

- *Suma de errores cuadráticos normalizada (NSSE)*: Una de las desventajas que se observan con el SSE es que es sensible al número de segmentos de la solución (este índice tiene a disminuir a medida que el número de segmentos decrece). Por este motivo, un enfoque más consistente sería dividir el error entre el número de segmentos:

$$NSSE = \frac{\frac{1}{N}\sum_{i=1}^{k}\sum_{\mathbf{x}\in\mathcal{C}_i} d(\mathbf{x}, \mathbf{c}_i)^2}{m} \tag{3.8}$$

  El *fitness* en este caso también se define como $f = \frac{1}{1+NSSE}$.

- *Índice Caliński and Harabasz(CH)*: Se ha demostrado que es un índice muy bueno para adjustar el valor de k. Se define como:

$$CH = \frac{\mathtt{Tr}(\mathbf{S}_B) \cdot (N-k)}{\mathtt{Tr}(\mathbf{S}_W) \cdot (k-1)}, \tag{3.9}$$

  donde N es el número de patrones, y $\mathtt{Tr}(\mathbf{S}_B)$ y $\mathtt{Tr}(\mathbf{S}_W)$ son las trazas de las matrices de dispersión entre y dentro de las clases, respectivamente. Como este índice tiene que ser maximizado el *fitness* se define como $f = CH$.

- *Índice Davies-Bouldin(DB)*: Este índice también trata de maximizar la distancia entre-*cluster* mientras que minimiza la inter-*cluster*. Se calcula con la expresión siguiente:

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{i\neq j} \frac{\alpha_i + \alpha_j}{d(\mathbf{c}_i, \mathbf{c}_j)}, \tag{3.10}$$

  siendo α_i la distancia media de todos los elementos en un *cluster*

$\mathcal{C}_i$ a su centroide $\mathbf{c}_i$, y $d(\mathbf{c}_i, \mathbf{c}_j)$ la distancia entre los centroides $\mathbf{c}_i$ y $\mathbf{c}_j$. Como este índice debe ser minimizado, la función de *fitness* se define como $f = \frac{1}{1+DB}$.

- *Índice Dunn (DU) y variantes*: Este índice trata de indentificar *clusters* que son compactos y están bien separados. En este caso, la distancia entre dos *clusters* se define como $\delta(\mathcal{C}_i, \mathcal{C}_j) = \text{mín}_{\mathbf{x}\in\mathcal{C}_i,\mathbf{y}\in\mathcal{C}_j}\, d(\mathbf{x}, \mathbf{y})$, es decir, la mínima distancia entre un par de puntos $\mathbf{x}$ e $\mathbf{y}$ pertenecientes a $\mathcal{C}_i$ y $\mathcal{C}_j$. Además, se define el diámetro $\texttt{diam}(\mathcal{C}_i)$ del *cluster* $\mathcal{C}_i$ como la distancia máxima entre dos de sus miembros: $\texttt{diam}(\mathcal{C}_i) = \text{máx}_{\mathbf{x},\mathbf{y}\in\mathcal{C}_i}\, d(\mathbf{x}, \mathbf{y})$. Luego el índice Dunn se construye como:

$$DU = \min_{i=1,\ldots,k} \left(\min_{j=i+1,\ldots,k} \left(\frac{\delta(\mathcal{C}_i, \mathcal{C}_j)}{\text{máx}_{l=1,\ldots,k}\, \texttt{diam}(\mathcal{C}_l)} \right) \right). \tag{3.11}$$

Se conoce que el índice Dunn es sensible al ruido, sin embargo, esta desventaja puede evitarse considerando diferentes definiciones de distancia de *cluster* o del diámetro de *cluster*. En este Trabajo Fin de Máster, se han considerado cuatro definiciones diferentes de este índice (correspondientes a definiciones diferentes de la distancia entre *clusters* y del diametro del *cluster*). Por ejemplo, como se propone en Xu and Wunsch [2008], el diámetro de un *cluster* se puede calcular como:

$$\texttt{diam}(\mathcal{C}_i) = \frac{1}{N_{\mathcal{C}_i}(N_{\mathcal{C}_i} - 1)} \sum_{\mathbf{x},\mathbf{y}\in\mathcal{C}_i} d(\mathbf{x}, \mathbf{y}), \tag{3.12}$$

donde $N_{\mathcal{C}_i}$ es el número de patrones pertenecientes al *cluster* $\mathcal{C}_i$. Esta estimación del diámetro de un *cluster* hace que el índice Dunn sea mucho más robusto al ruido, por lo que ha sido incluida en los experimentos. También se han incluido tres variantes de este índice que producen los mejores resultados en Arbelaitz et al. [2013], que son GD33, GD43 y GD53, que se corresponden

con la variaciones siguientes de $\delta(\mathcal{C}_i, \mathcal{C}_j)$:

$$\delta^3(\mathcal{C}_i, \mathcal{C}_j) = \frac{1}{N_{\mathcal{C}_i} N_{\mathcal{C}_j}} \sum_{\mathbf{x} \in \mathcal{C}_i} \sum_{\mathbf{y} \in \mathcal{C}_j} d(\mathbf{x}, \mathbf{y}), \tag{3.13}$$

$$\delta^4(\mathcal{C}_i, \mathcal{C}_j) = d(\overline{\mathcal{C}_i}, \overline{\mathcal{C}_j}), \tag{3.14}$$

$$\delta^5(\mathcal{C}_i, \mathcal{C}_j) = \frac{1}{N_{\mathcal{C}_i} + N_{\mathcal{C}_j}} \left(\sum_{\mathbf{x} \in \mathcal{C}_i} d(\mathbf{x}, \overline{\mathcal{C}_i}) + \sum_{\mathbf{y} \in \mathcal{C}_j} d(\mathbf{y}, \overline{\mathcal{C}_j}) \right), \tag{3.15}$$

donde $\overline{\mathcal{C}_i}$ representa la media del *cluster*. El término $\texttt{diam}(\mathcal{C}_i)$ se define en este caso como:

$$\texttt{diam}(\mathcal{C}_i) = \frac{2}{N_{\mathcal{C}_i}} \sum_{\mathbf{x} \in \mathcal{C}_i} d^*_{ps}(\mathbf{x}, \mathcal{C}_i), \tag{3.16}$$

donde $d^*_{ps}(\mathbf{x}, \mathcal{C}_i)$ es la distancia *Point Symmetry-Distance* entre el objeto $\mathbf{x}$ y el *cluster* $\mathcal{C}_i$, definida en Bandyopadhyay and Saha [2008]. Como todas las variaciones de este índice deben ser maximizadas, el *fitness* se define como $f = DU$.

- *Índice Silhouette (SI)*: Para este índice, la cohesión se calcula como la distancia entre todos los puntos de un mismo *cluster*, y la separación se mide usando la distancia del vecino más cercano. Se define como:

$$SI = \frac{1}{N} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{C}_i} \frac{b(\mathbf{x}, \mathcal{C}_i) - a(\mathbf{x}, \mathcal{C}_i)}{\text{máx}(a(\mathbf{x}, \mathcal{C}_i), b(\mathbf{x}, \mathcal{C}_i))}, \tag{3.17}$$

donde a y b se corresponde respectivamente a las distancias intra-*cluster* e inter-*cluster* y se definen como sigue:

$$a(\mathbf{x}, \mathcal{C}_i) = \frac{1}{N_{\mathcal{C}_i}} \sum_{y \in \mathcal{C}_i} d(\mathbf{x}, \mathbf{y}), \tag{3.18}$$

$$b(\mathbf{x}, \mathcal{C}_i) = \min_{\mathcal{C}_l, l \neq i} \left\{ \frac{1}{N_{\mathcal{C}_l}} \sum_{\mathbf{y} \in \mathcal{C}_l} d(\mathbf{x}, \mathbf{y}) \right\}. \tag{3.19}$$

- *Índice COP (COP)*: La cohesión es estimada, en este índice, usando la distancia de los puntos a su centroide y la distancia al vecino más lejano. Se define como:

$$COP = \frac{1}{N} \sum_{i=1}^{k} \frac{\sum_{\mathbf{y} \in \mathcal{C}_k} d(\mathbf{y}, \mathbf{c}_k)}{N_{\mathcal{C}_k} \cdot \min_{\mathbf{x} \notin \mathcal{C}_k} \max_{\mathbf{y} \in \mathcal{C}_k} d(\mathbf{x}, \mathbf{y})}. \tag{3.20}$$

3.4.2. Función de *Fitness* para el Objetivo de Reducir el Error de la Aproximación

El propósito de esta segunda función de *fitness* es ayudar al algoritmo multiobjetivo evolutivo a reducir el error entre la aproximación proporcionada por la segmentación y la serie original. Cabe destacar que la aproximación será estimada mediante la función de regresión con el grado del polinomio que se haya utilizado en la función de *fitness* referente al *clustering*. La función de *fitness* se puede definir por tanto, como la minimización de la diferencia entre cada punto real de la serie temporal y su correspondiente aproximación. Así pues, el error del i-ésimo punto en el cromosoma c puede definirse como $e_i(c) = y_i - \bar{y}_i(c)$, donde y_i es el valor real de la serie temporal, e $\bar{y}_i(c)$ es el valor de la aproximación de un punto i del cromosoma c. Para este objetivo, se han implementado tres índices:

- El primer índice es la raiz cuadrada del error cuadrático medio (*Root Mean Squared Error, RMSE*), y es calculado de la siguien-

te forma:

$$RMSE(c) = \sqrt{MSE_c} = \sqrt{\frac{1}{m}\sum_{s=1}^{m} MSE_{c,s}} = \quad (3.21)$$
$$= \sqrt{\frac{1}{m}\sum_{s=1}^{m} \frac{1}{t_s - t_{s-1}+1}\sum_{i=t_{s-1}}^{t_s} e_i^2(c)}$$

donde $MSE_{c,s}$ es el Error Cuadrático Medio (*Mean Squared Error, MSE*) de cada segmento s, y MSE_c es el valor medio de todos los $MSE_{c,s}$.

- El segundo índice (denominado RMSEp, error ponderado a nivel de puntos) es obtenido mediante la siguiente expresión:

$$RMSEp(c) = \sqrt{\frac{1}{N}SSE(c)} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2(c)} \quad (3.22)$$

donde podemos ver que se utiliza la suma de los errores cuadráticos (*SSE(c)*), la cual no promedia los errores por segmentos.

- El último índice implementado es el llamado MAXe, y se obtiene mediante la siguiente fórmula:

$$MAXe(c) = \max_{i=1}^{N}(e_i^2(c)) \quad (3.23)$$

donde máx representa a la función máximo. Por tanto, esté índice trata de minimizar el error máximo cometido por la aproximación.

Como los tres índices tienen que ser minimizados, la función de *fitness* se define como $f = \frac{1}{1+RMSE}$, $f = \frac{1}{1+RMSEp}$ y $f = \frac{1}{1+MAXe}$, respectivamente.

3.5. Selección

Algunas de las técnicas de selección aplicadas a los algoritmos evolutivos son:

- Selección directa: toma individuos de acuerdo a un criterio objetivo, como son "los x mejores", "los x peores"... Son empleados con mucha frecuencia cuando se quieren seleccionar dos individuos distintos, y se selecciona el primero por un método aleatorio o estocástico.
- Selección aleatoria: puede ser realizada por selección equiprobable o selección estocástica.
- Selección neutra: selecciona los individuos en el orden que le llegan, sin realizar ningún tipo de procesamiento sobre ellos.
- Selección por torneos: realizar el proceso de selección mediante torneos (un torneo por cada uno de los individuos que haya que seleccionar). El tamaño del torneo es variable.
- Selección por ruleta: selecciona individuos utilizando una ruleta, en la que el área correspondiente a cada individuo es proporcional a su aptitud.

En el NSGA2 clásico, la técnica usada de selección es el torneo, y usualmente de tamaño dos. En el presente Trabajo Fin de Máster, además de este tipo de selección se ha decidido añadir la selección directa de todos los individuos (dando la posibilidad al usuario de usar la que desee). Es decir, en cada generación, todos los individuos de la población serán destinados a reproducirse. Esta decisión se ha tomado ya que al hacer uso del torneo, hemos observado que, en problemas de segmentación de series temporales, el algoritmo converge prematuramente, produciéndose poblaciones con una gran cantidad de cromosomas repetidos, y mediante esta selección se favorece la diversidad.

3.6. Operador de Cruce

El algoritmo ha sido dotado de un operador de cruce cuya función primordial es realizar una explotación de las soluciones existentes

hasta el momento. Para cada padre, el operador se aplica con una probabilidad p_c. El operador aleatoriamente selecciona otro padre y un punto de cruce. El cruce consiste en el intercambio de la partes izquierda y derecha entre ambos padres con respecto al punto de cruce.

Después de este paso, el algoritmo detecta si las soluciones son factibles. Esto se refiere a que las soluciones deben poseer al menos el tamaño mínimo de segmento, ya que podría no ser posible el cálculo posterior de algunas métricas (por ejemplo, si el grado del polimonio es 2 y tenemos alguna solución que presenta un segmento con dos puntos, existirían infinitas parábolas válidas y, por tanto, infinitos coeficientes). Si el algoritmo detecta que alguna solución no es válida, se vuelve a aplicar el cruce entre ambos padres, hasta un máximo de tres intentos. Si el cruce no se realiza con éxito, el segundo padre es sustituido por otro al azar y se comienza a probar el cruce de nuevo. Este proceso se repite hasta que el operador de cruce termina satisfactoriamente.

Un ejemplo se puede observar en la Figura 3.2.

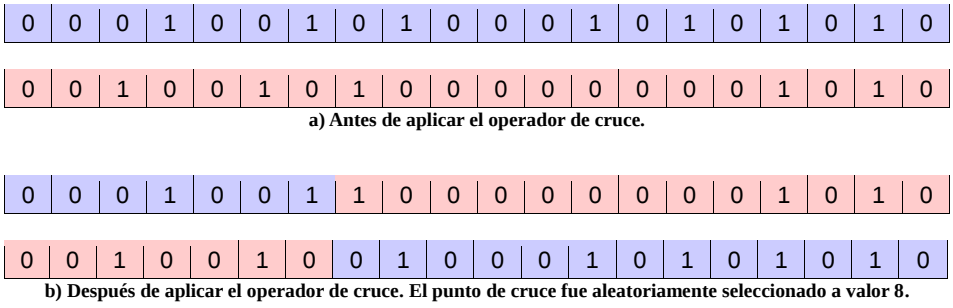


Figura 3.2: Operador de cruce del algoritmo multiobjetivo evolutivo. a) Antes de aplicar el operador de cruce. b) Después de aplicar el operador de cruce. El punto de cruce fue aleatoriamente seleccionado a valor 8.

3.7. Operador de Mutación

Dos operadores de mutación han sido incluidos en el algoritmo con el objetivo de reducir la dependencia de las soluciones de la población

inicial y escapar de óptimos locales. La probabilidad p_m es decidida por el usuario. Una vez que un cromosoma ha sido seleccionado para ser mutado, el tipo de mutación se selecciona aleatoriamente de entre las dos siguientes:

1. Añadir o eliminar (con la misma probabilidad) puntos de corte en el cromosoma.

2. Desplazar puntos de corte hacia la derecha o izquierda (con la misma probabilidad).

Para todas las mutaciones, el número de puntos de corte a mutar es decidido por el usuario mediante un parámetro que representa un porcentaje del número de puntos de corte actual de la solución. Cuando se decide que el mutador debe ser el desplazamiento de los puntos de corte, cada uno de los seleccionados se mueve en el intervalo entre el punto de corte anterior o el posterior (según la dirección del desplazamiento), teniendo en cuenta que se debe respetar el tamaño mínimo de segmento. Si el mutador no produce una solución factible, se usa el mismo procedimiento que en el operador de cruce.

Un ejemplo se puede observar en la Figura 3.3.

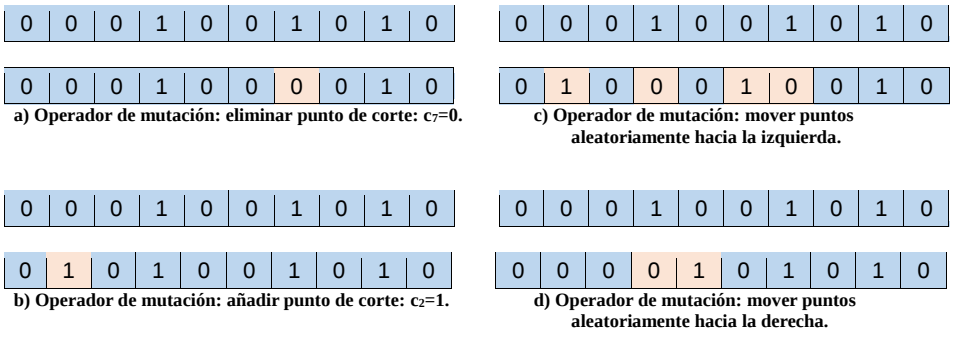


Figura 3.3: Operador de mutación del algoritmo multiobjetivo evolutivo. a) Operador de mutación: eliminar punto de corte $c_7 = 0$. b) Operador de mutación: añadir punto de corte $c_2 = 1$. c) Operador de mutación: mover puntos aleatoriamente hacia la izquierda. d) Operador de mutación: mover puntos aleatoriamente hacia la derecha.

3.8. Reemplazo

Una vez que el algoritmo ha seleccionado los padres, ha ejecutado el operador de cruce y el de mutación, y se han evaluado las soluciones desde los dos puntos de vista (función de *fitness* de *clustering* y función de *fitness* de error), es el momento de decidir que elementos pasan a la siguiente generación.

Para ello, se hace uso de los métodos de ordenación por no dominancia y el operador de comparación por *crowding* explicados en la sección 2.5. De esta forma, se ha implementado una función que combina ambos procesos y devuelve la población ordenada por frentes, y dentro de cada frente por diversidad *crowding*.

Así pues, para la siguiente generación sobrevivirán los N individuos mejor situados en el ranking anterior.

3.9. Criterio de Parada

La condición de parada consiste en alcanzar un determinado número de generaciones.

3.10. Pseudocódigo del Algoritmo

```
Algoritmo F = GMOTSS(T)
  Generar una población aleatoria de tamaño N
  Evaluar todas las segmentaciones de la población inicial
      usando las dos funciones de fitness
  Realizar un ranking de las soluciones iniciales mediante
      la ordenación por no dominancia y ordenar los
     individuos dentro de cada frente mediante el
     operador de comparación crowding
  Mientras no se satisfaga la condición de parada
    Seleccionar los padres para reproducirse
    Generar la población de hijos: Aplicar los operadores
        de cruce y mutación, en este orden
    Evaluar la población de hijos usando las dos funciones
        de fitness
    Unir la población de padres e hijos
    Realizar un ranking de la población resultante de la
        unión mediante la ordenación por no dominancia y
        ordenar los individuos dentro de cada frente
        mediante el operador de comparación crowding
  FinMientras
  Devolver la población de segmentaciones de series
     temporales y la mejor segmentación en cada fitness
Fin
```

4 Resultados Experimentales

En esta sección, se recogen los resultados experimentales del algoritmo propuesto. A continuación, se presentan las bases de datos utilizadas, la configuración de tres experimentos, y sus resultados.

4.1. Bases de Datos Utilizadas

En este Trabajo Fin de Máster, se ha analizado el rendimiento del algoritmo en cuatro series temporales, de ámbito muy diferente, para testear su adaptabilidad en diferentes áreas con el fin de realizar un algoritmo genérico. Las series temporales utilizadas son las siguientes:

- La primera serie temporal utilizada es el IBEX35. Es uno de lo índices bursátiles oficiales de Madrid, el cual está compuesto de los valores más volátiles que cotizan en el *Computer Assisted Trading System*. En nuestro estudio experimental, se han considerado los precios de cierre desde el 4 de enero de 1992 hasta el 26 de Septiembre de 2014, presentando un total de 5730 observaciones (*ver https://es.finance.yahoo.com/*). La representación completa de la serie se puede ver en la Figura 4.1 (a).

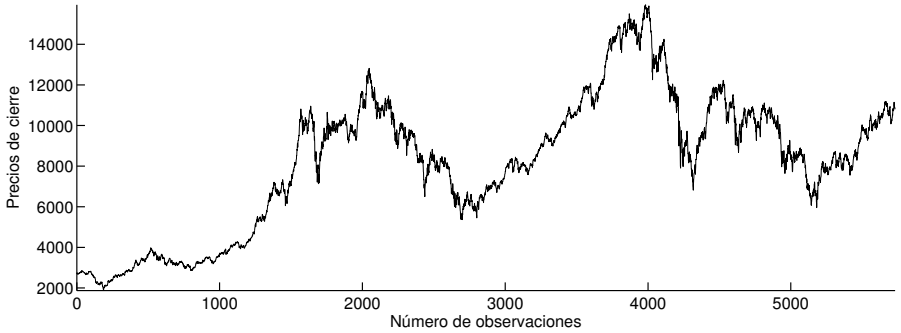
- La segunda serie temporal ha sido extraída de un *benchmark*

utilizado en investigaciones de redes neuronales y aprendizaje de máquina [Donoho and Johnstone, 1994, Donoho et al., 1995, Donoho and Johnstone, 1995]. Este *benchmark* posee muchas entradas, dependiendo de la cantidad de ruido añadido, o bien, de la presencia de linealidad. Concretamente, los *benchmarks Donoho-Johnstone* consisten en cuatro funciones, llamadas *Blocks, Bumps, Heavisine, y Doppler*, a las que se les añade ruido aleatorio para producir un infinito número de conjuntos de datos. En nuestros experimentos, se ha usado la función *Blocks*, con una cantidad de ruido medio, presentando un total de 2048 observaciones. Esta serie temporal puede ser descargada en *https://sites.google.com/site/icdmmdl/*. La serie completa puede verse en la Figura 4.1 (b).

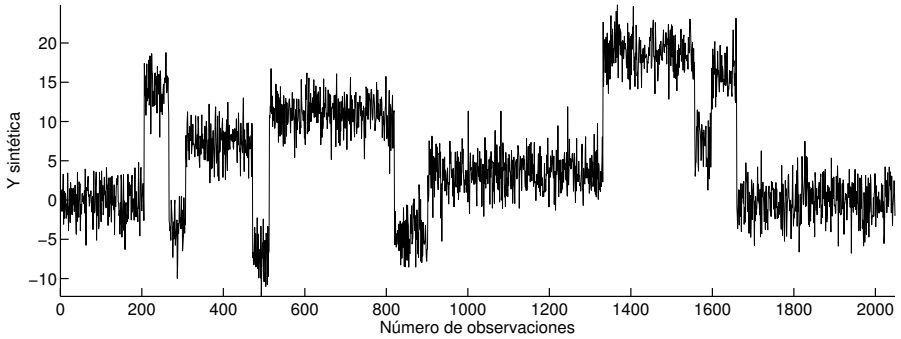
- La tercera serie temporal está referida a la altura significativa de ola, registrada por una boya del Golfo de Alaska: específicamente, es una boya contenida en el *National Buoy Center of the USA* [NDBC, 2015], con el número de identificación 46001. En la fase experimental propuesta, se han considerado los datos contenidos entre el 1 de Enero de 2008 hasta el 31 de Diciembre de 2013, tomando valores con una resolución de 6 horas, es decir, un valor de la boya cada 6 horas. La serie completa se presenta en la Figura 4.1 (c).

- La última serie temporal es un conjunto de datos de electrocardiogramas recogidos por el *PhysioBank ATM* de la base de datos *MIT BIH Arrhythmia* [Moody and Mark, 2001, Goldberger et al., 2000]. Para probar el algoritmo en esta base de datos, se ha utilizado la señal 108, específicamente, la señal MLII. Esta serie temporal es la más extensa de todos nuestros experimentos, con un total de 9000 observaciones. La Figura 4.1 (d) muestra la representación de la serie.

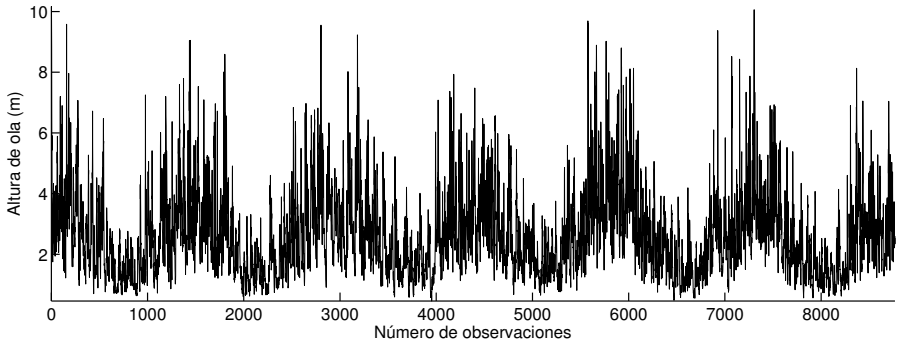
Como se puede observar, estas series temporales son muy diferentes tanto en longitud como en ámbito de aplicación.



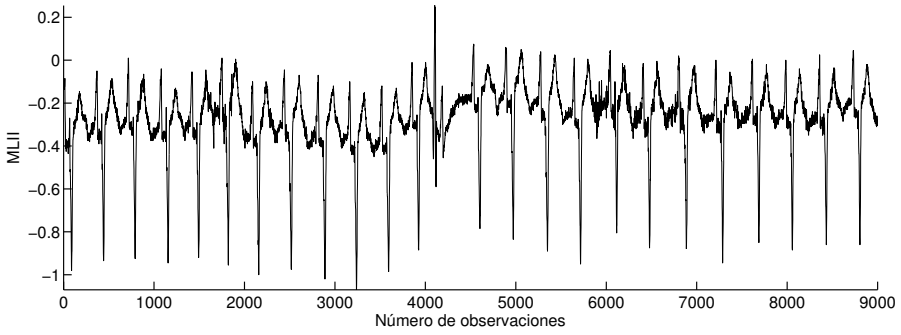
(a) IBEX35



(b) *Donoho-Jonhstone*



(c) Boya 46001



(d) *MIT BIH Arrhythmia*

Figura 4.1: Series temporales utilizadas en los experimentos

4.2. Experimento 1

En este experimento, se va a corroborar la hipótesis de partida de que el problema abordado se trata de un problema multiobjetivo en el que ambos objetivos son contrapuestos. Para ello, se ha decidido lanzar el algoritmo presentado en el capítulo 3, para todas las bases de datos, con la combinación de todos los *fitness* de *clustering* (DB, DU, CH, SSE, NSSE, SI, GD33, GD43, GD53 y COP, definidos en la sección 3.4.1), con los *fitness* de minimización del error (RMSE, RMSEp y MAXe, definidos en la sección 3.4.2).

4.2.1. Configuración de los Parámetros

Para este experimento, la configuración de los parámetros ha sido:

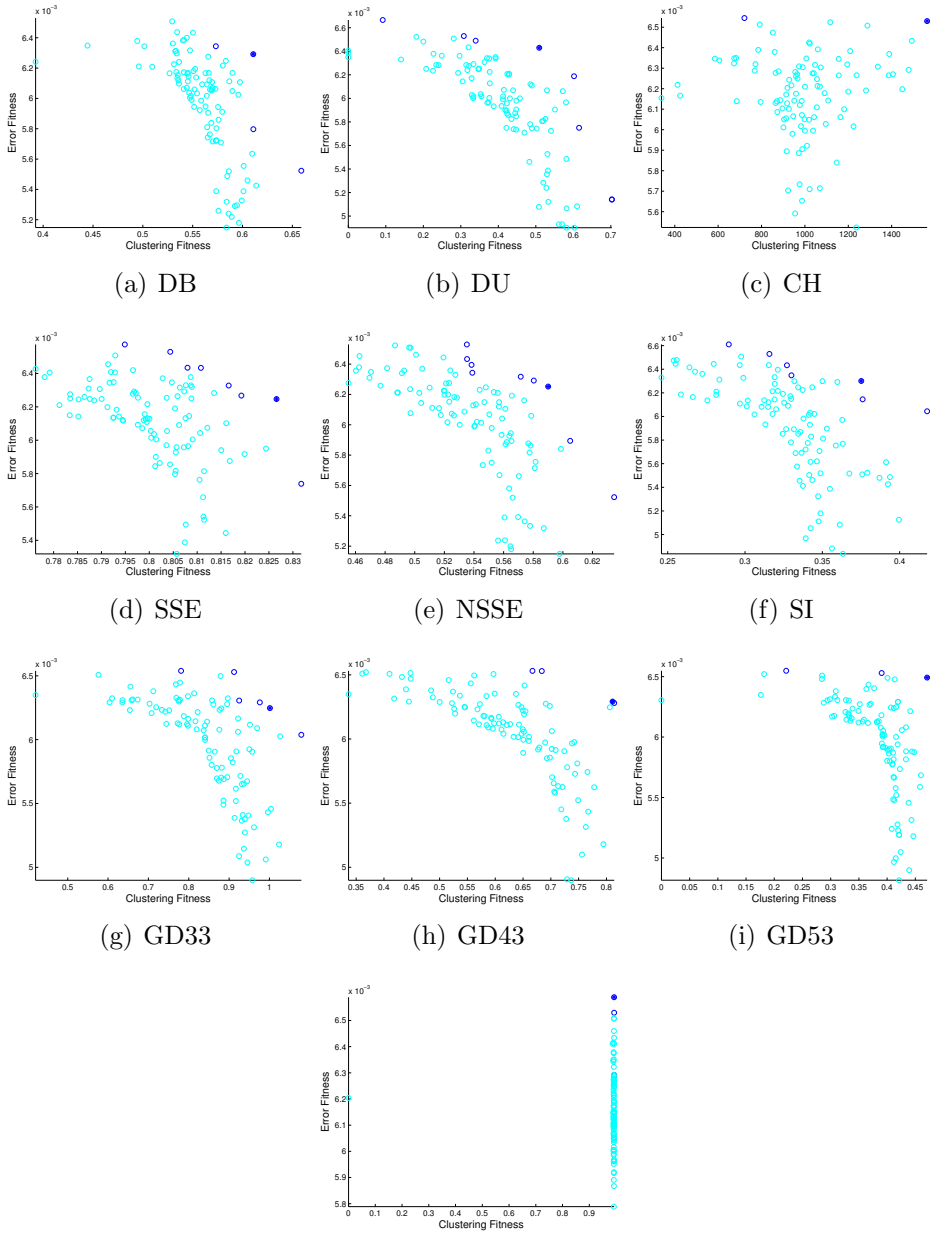
- El tamaño mínimo de segmento es $min_{size} = U(4, 20)$, siendo $U(x, y)$ un número aleatorio uniforme entre x e y.
- El tamaño máximo de segmento es $max_{size} = U(70, 120)$.
- El grado del polinomio es $n = 2$.
- El número de generaciones es $g = 3$.
- El tamaño de la población es $P = 100$.
- La probabilidad de cruce es $p_c = 0{,}8$.
- La probabilidad de mutación es $p_m = 0{,}2$.
- El porcentaje de puntos a ser mutados es del 20 %.
- El número de *clusters* se ha establecido en $k = 5$.
- El número de iteraciones para el *k-means* es $it = 20$.

4.2.2. Resultados y Discusión

Las Figuras 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, y 4.13, muestran los frentes de Pareto obtenidos en la última generación del algoritmo evolutivo multiobjetivo. Se puede ver cómo todas las métricas son contrapuestas: si se observa, cuanto más nos acercamos al óptimo en un *fitness* más se empeora en el otro. Por tanto, queda demostrado que el problema abordado es un problema de optimización multiobjetiva, en el que las métricas a optimizar son contrapuestas.

Aunque a simple vista cualquier métrica de error con cualquier índice de *clustering* puede ser adecuada, el RMSE es una métrica de error que minimiza error medio entre todos los segmentos. Esto provoca que, si existen segmentos pequeños muy refinados, los segmentos con una gran longitud puedan cometer un error muy grande, produciéndose aproximaciones buenas en términos de este *fitness*, pero que realmente son "engañosas". A raíz de observar este problema, el autor de este Trabajo Fin de Máster ha tratado este inconveniente con más detalle en un artículo publicado en el Congreso *Hybrid Artificial Intelligent Systems, HAIS 2016*, realizado en paralelo y que es adjuntado en el Apéndice A. Por este motivo, en el siguiente experimento se usará el RMSEp (ver sección 3.4.2) como *fitness* de minimización del error.

Una vez se ha decidido el *fitness* anterior en la aproximación, se ha considerado el uso del índice Dunn como *fitness* de *clustering*, ya que en las gráficas se observa que son dos métricas bien contrapuestas.



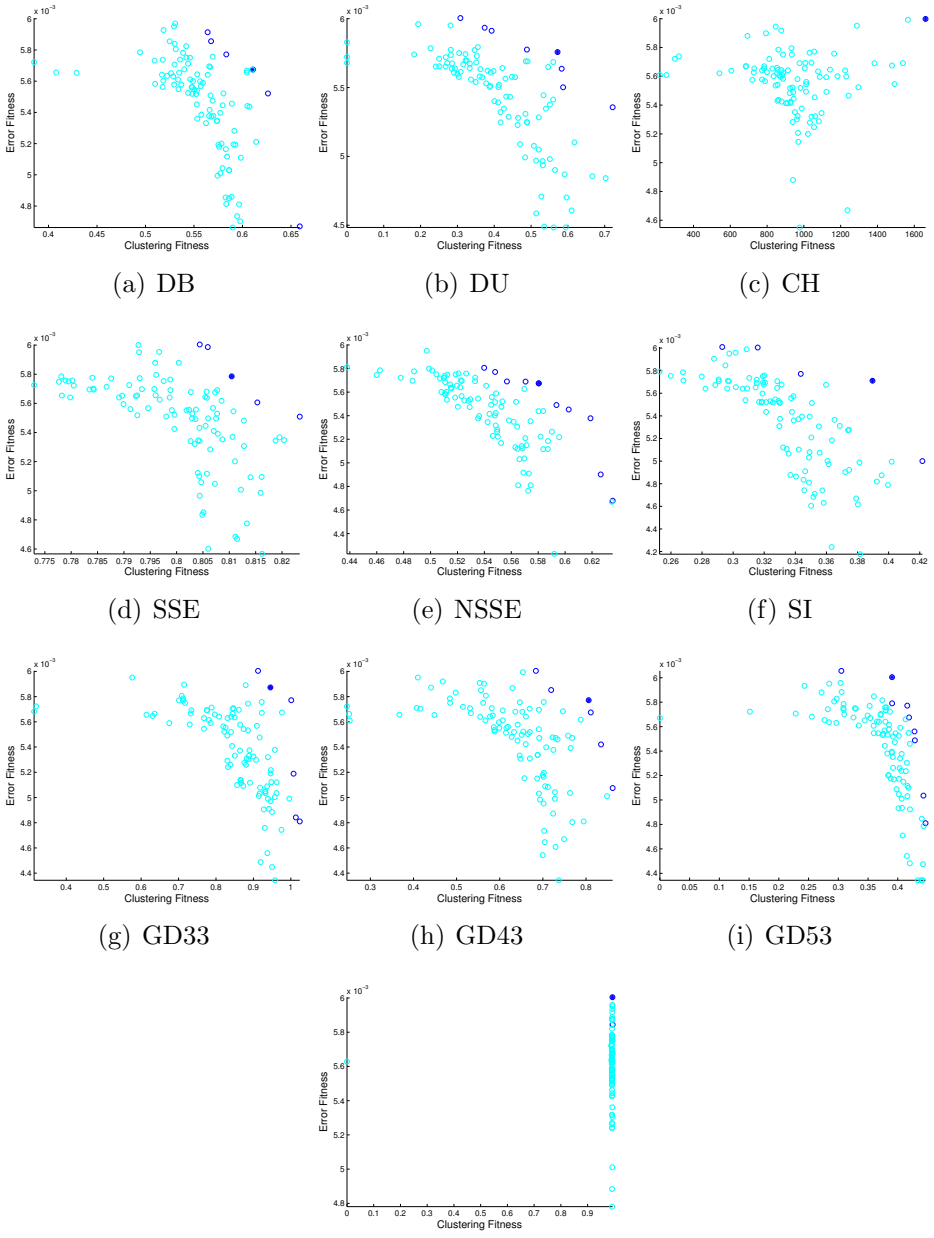
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.2: Frentes de Pareto para la serie Ibex: RMSE vs *Clustering Fitness*.



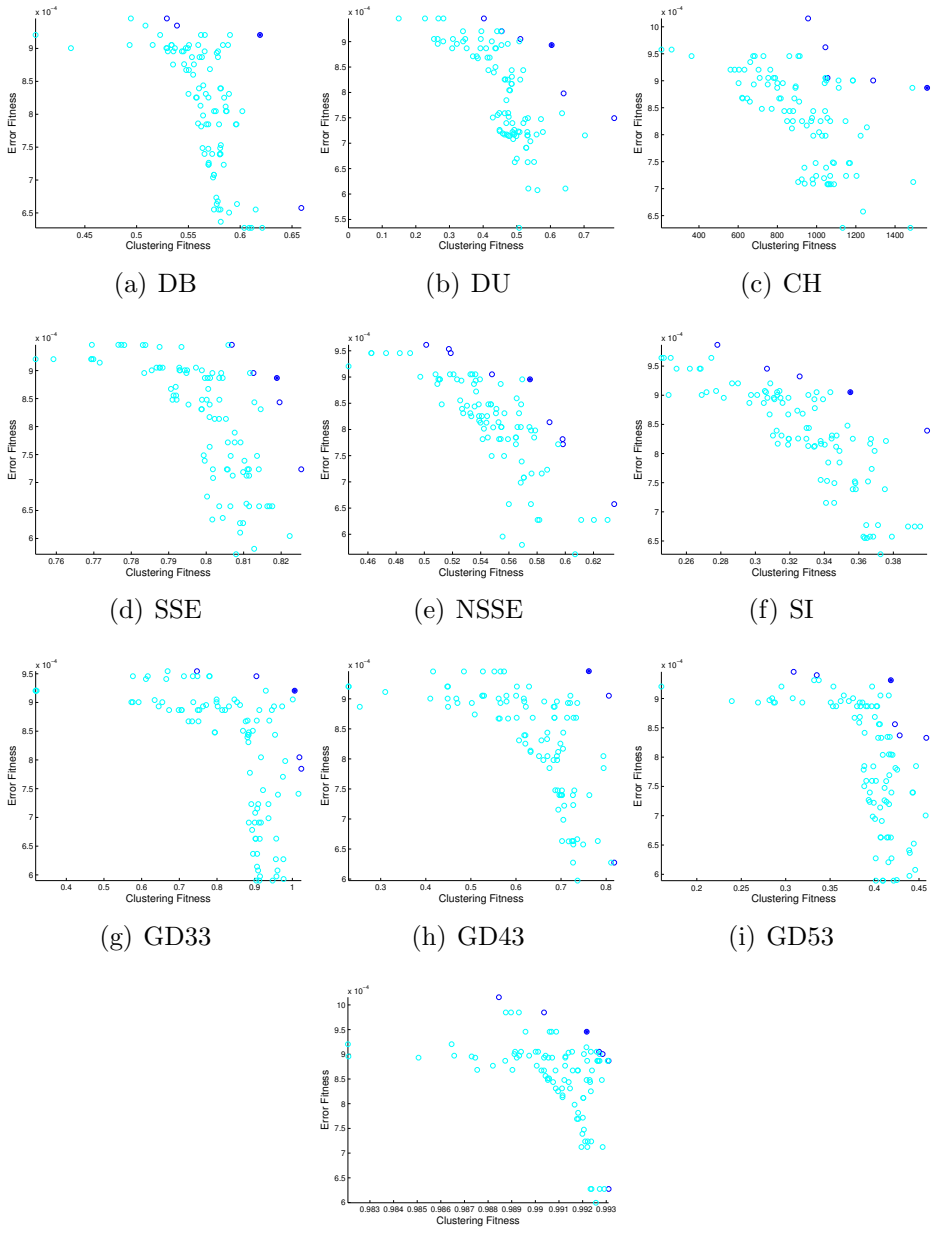
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.3: Frentes de Pareto para la serie Ibex: RMSEp vs *Clustering Fitness*.



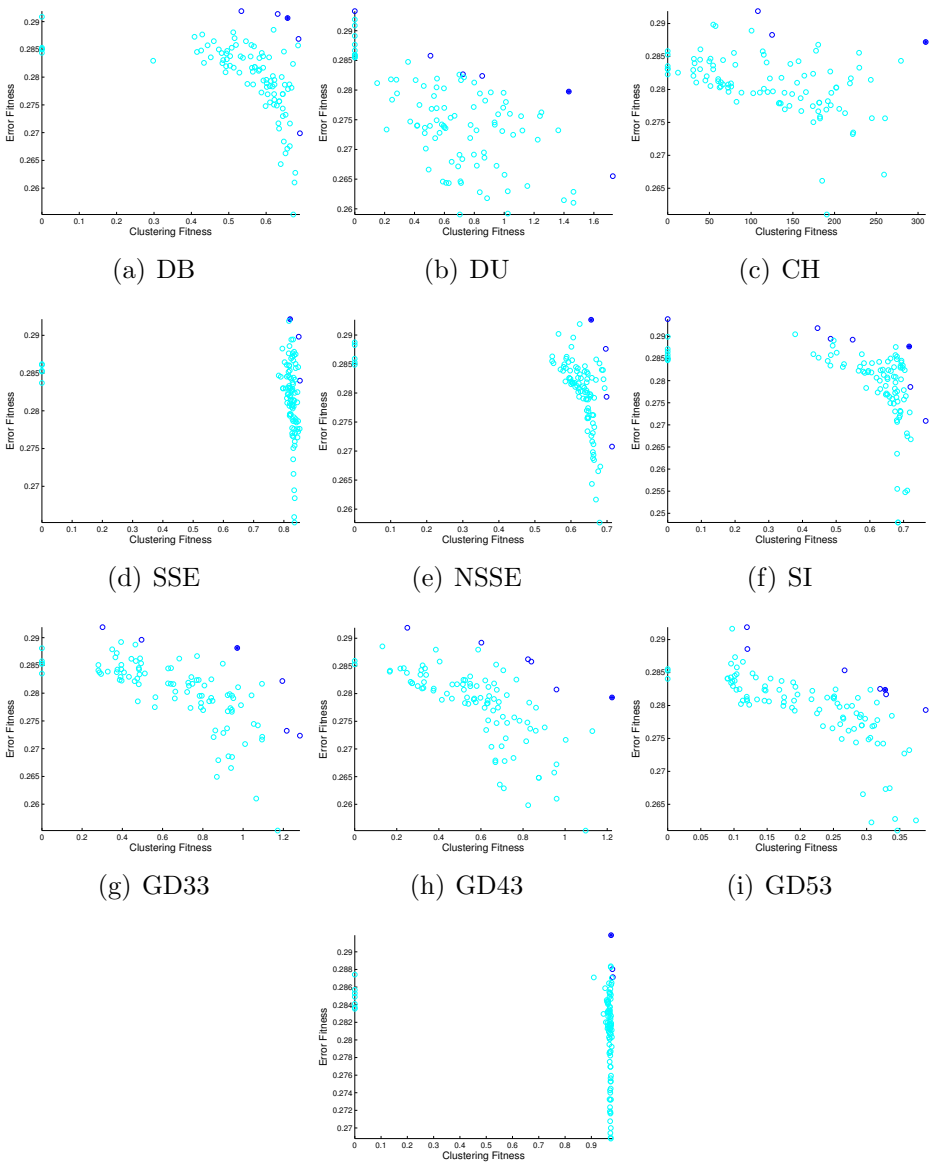
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.4: Frentes de Pareto para la serie Ibex: MAXe vs *Clustering Fitness*.



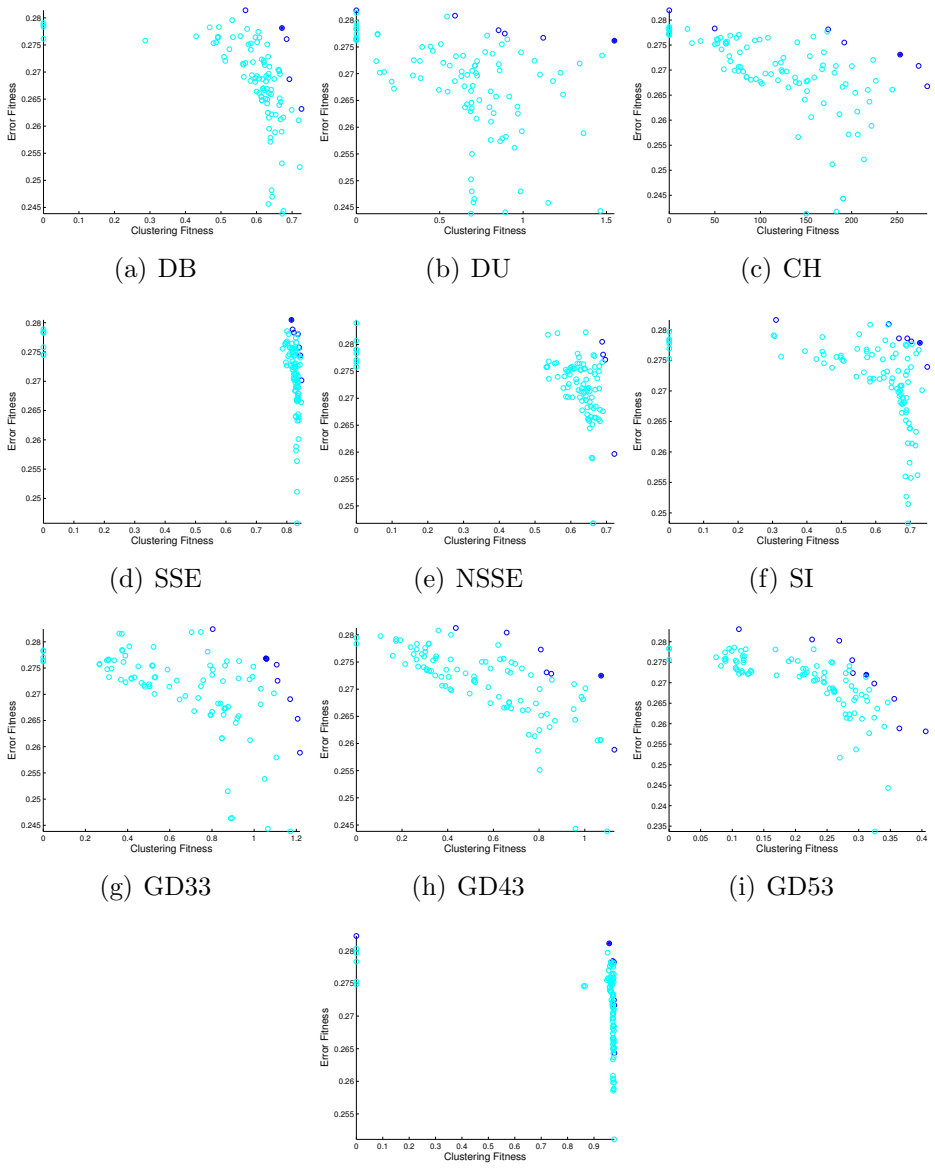
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.5: Frentes de Pareto para la serie Donoho-Johnstone: RMSE vs *Clustering Fitness*.



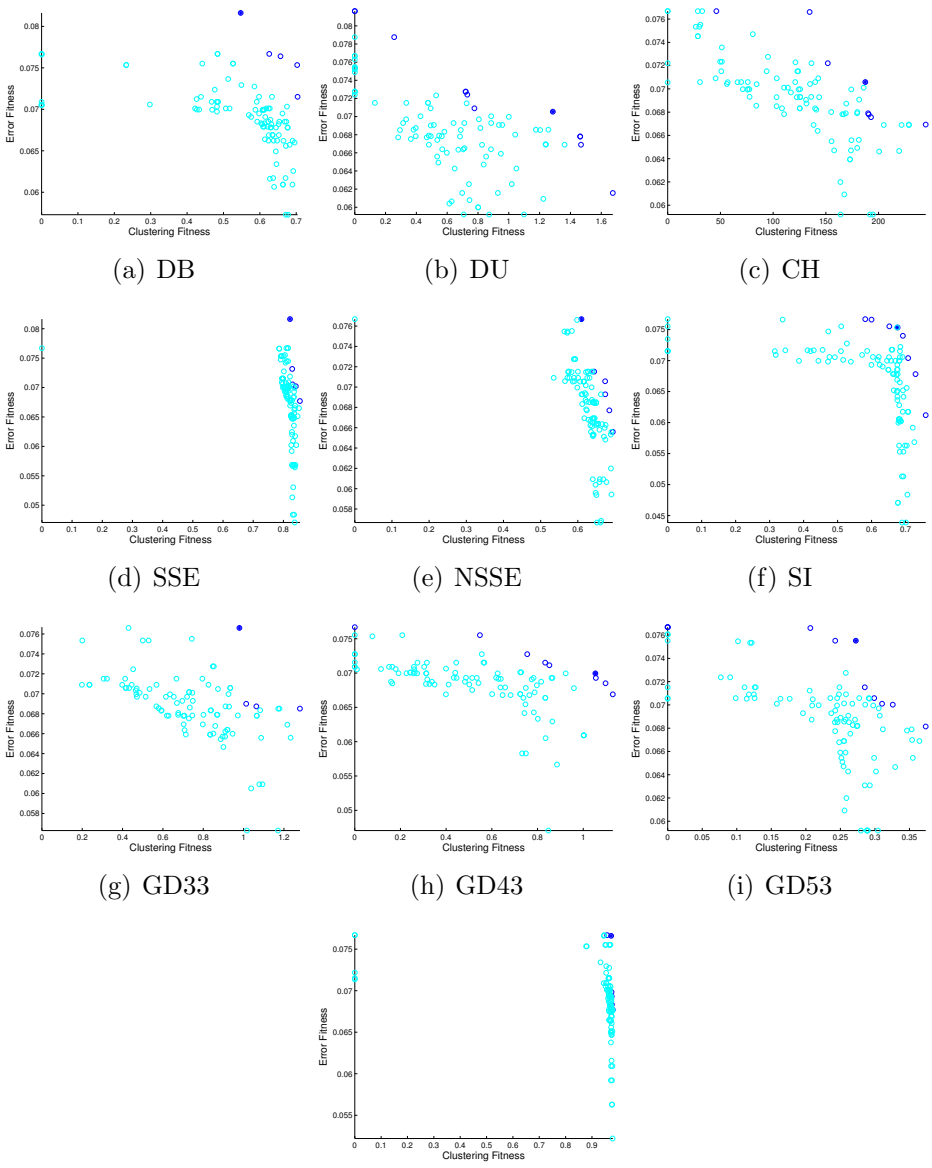
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.6: Frentes de Pareto para la serie Donoho-Johnstone: RMSEp vs *Clustering Fitness*.



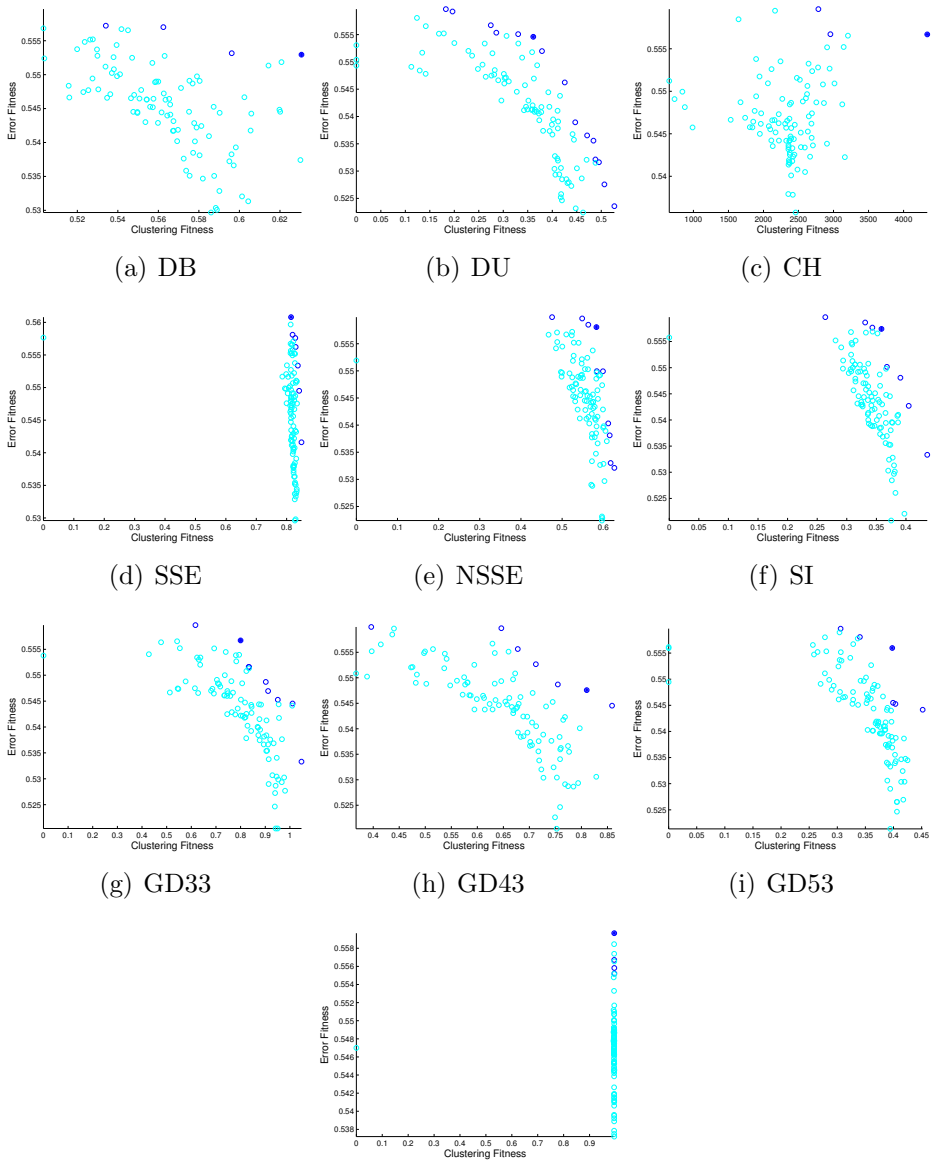
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.7: Frentes de Pareto para la serie Donoho-Johnstone: MAXe vs *Clustering Fitness*.



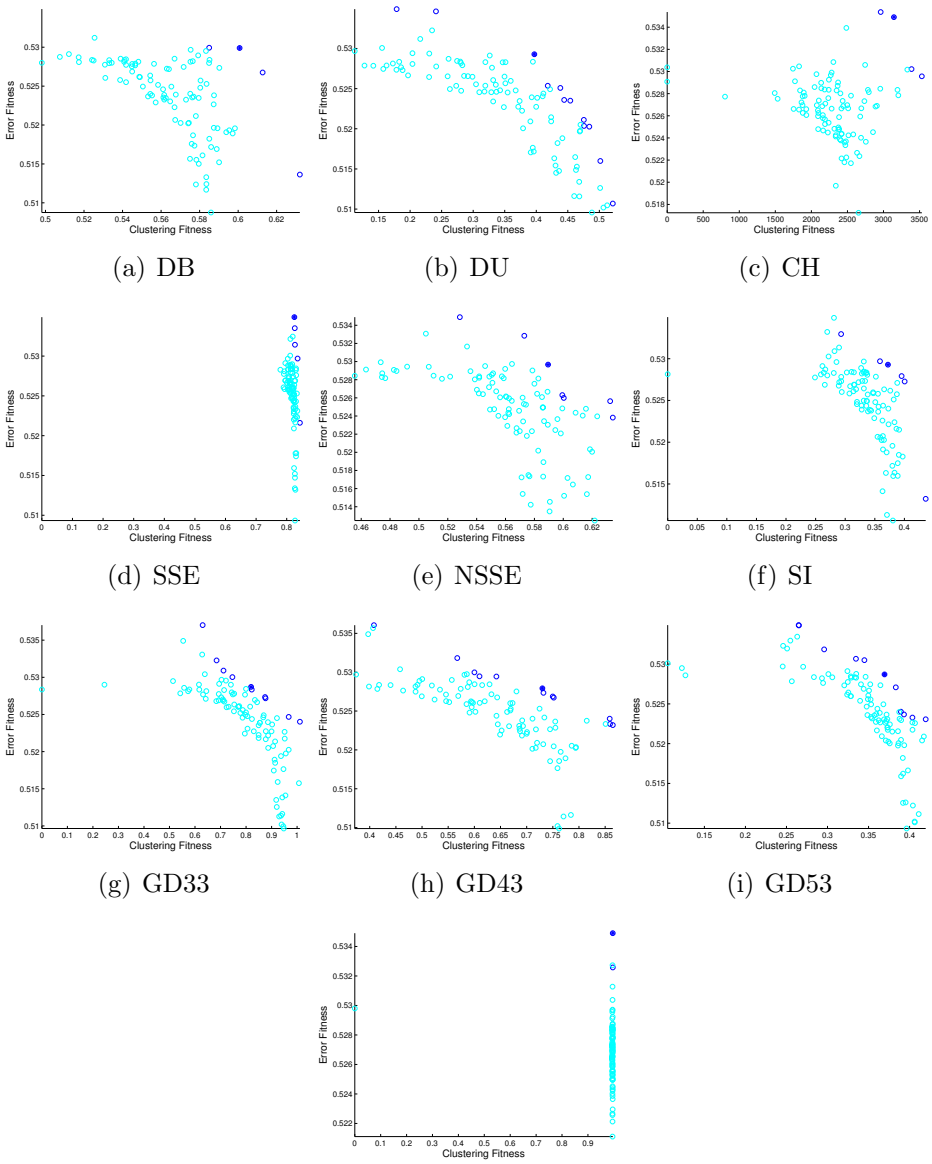
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.8: Frentes de Pareto para la serie Boya46001: RMSE vs *Clustering Fitness*.



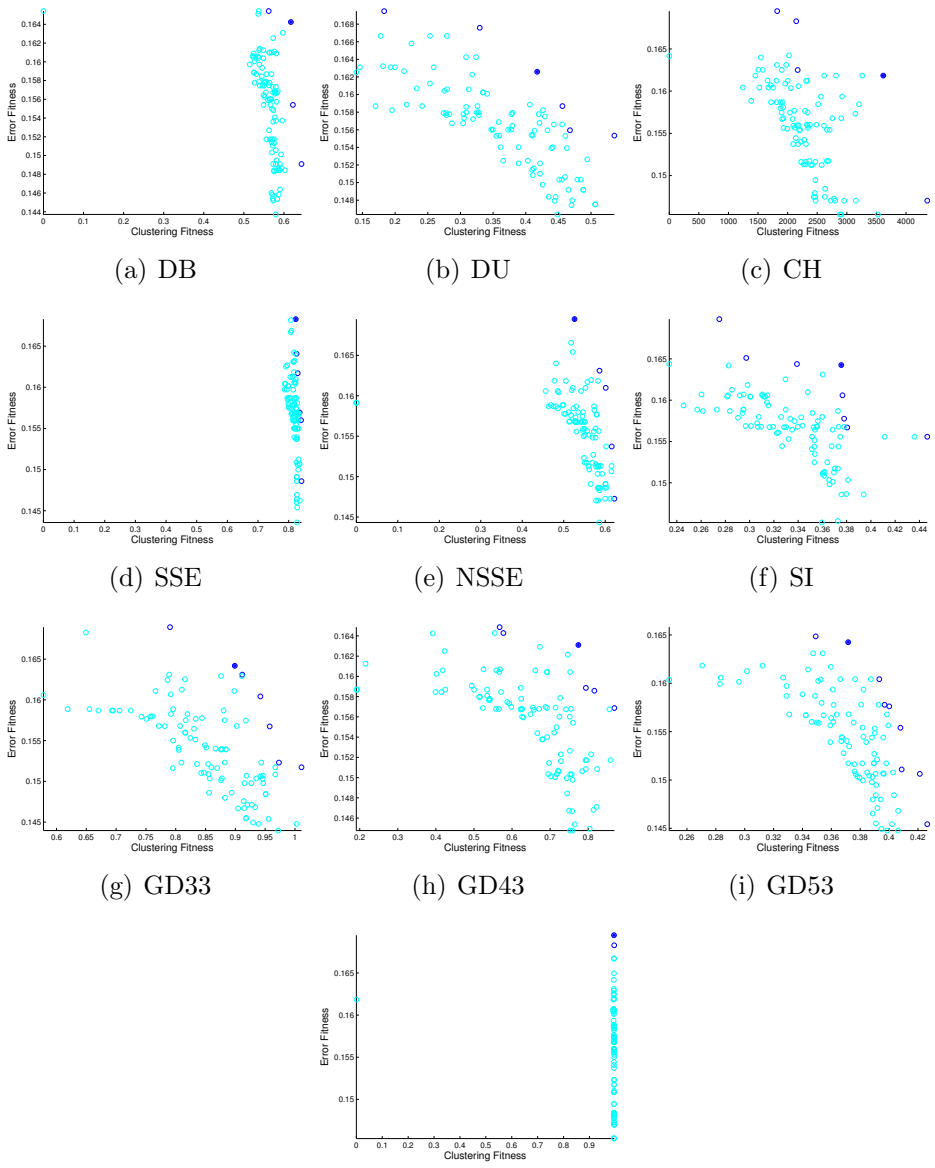
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.9: Frentes de Pareto para la serie Boya46001: RMSEp vs *Clustering Fitness*.



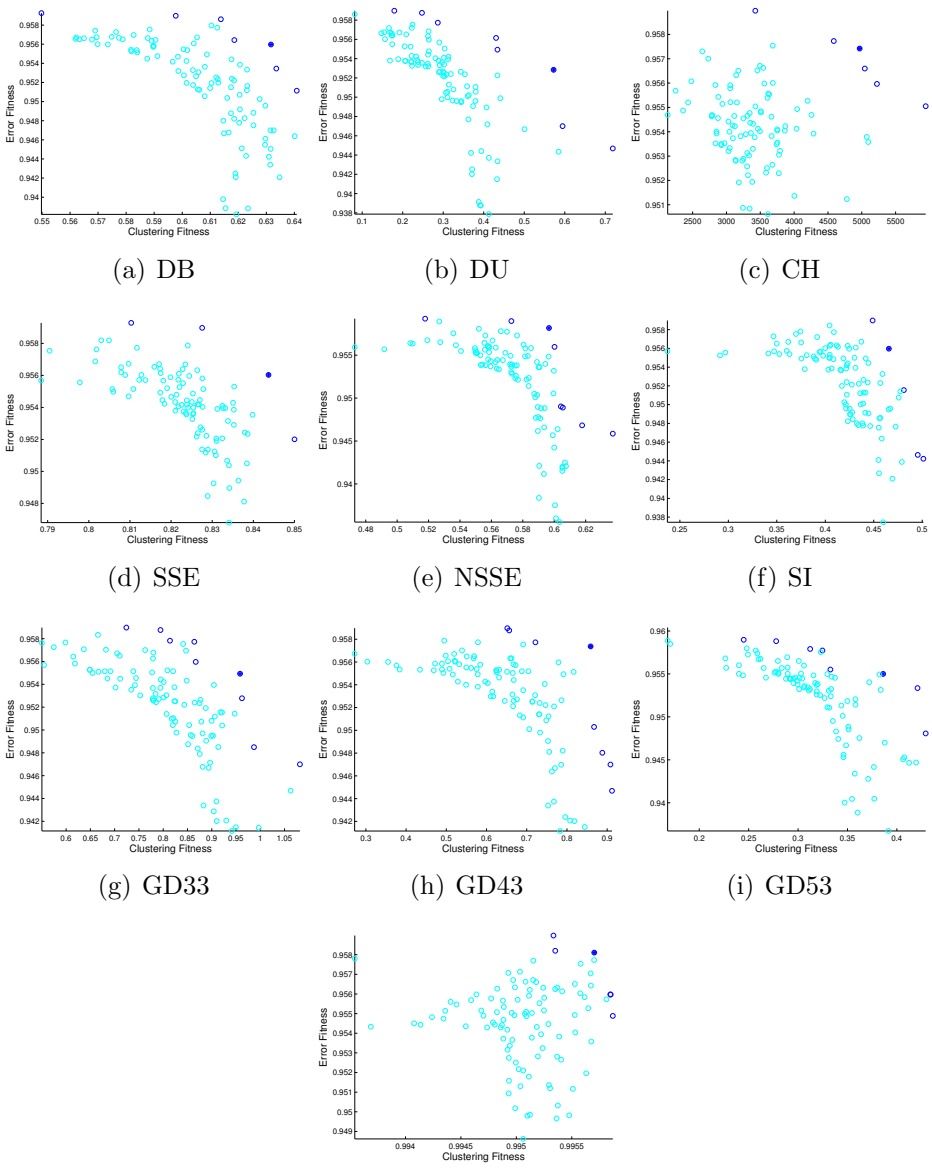
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.10: Frentes de Pareto para la serie Boya46001: MAXe vs *Clustering Fitness*.



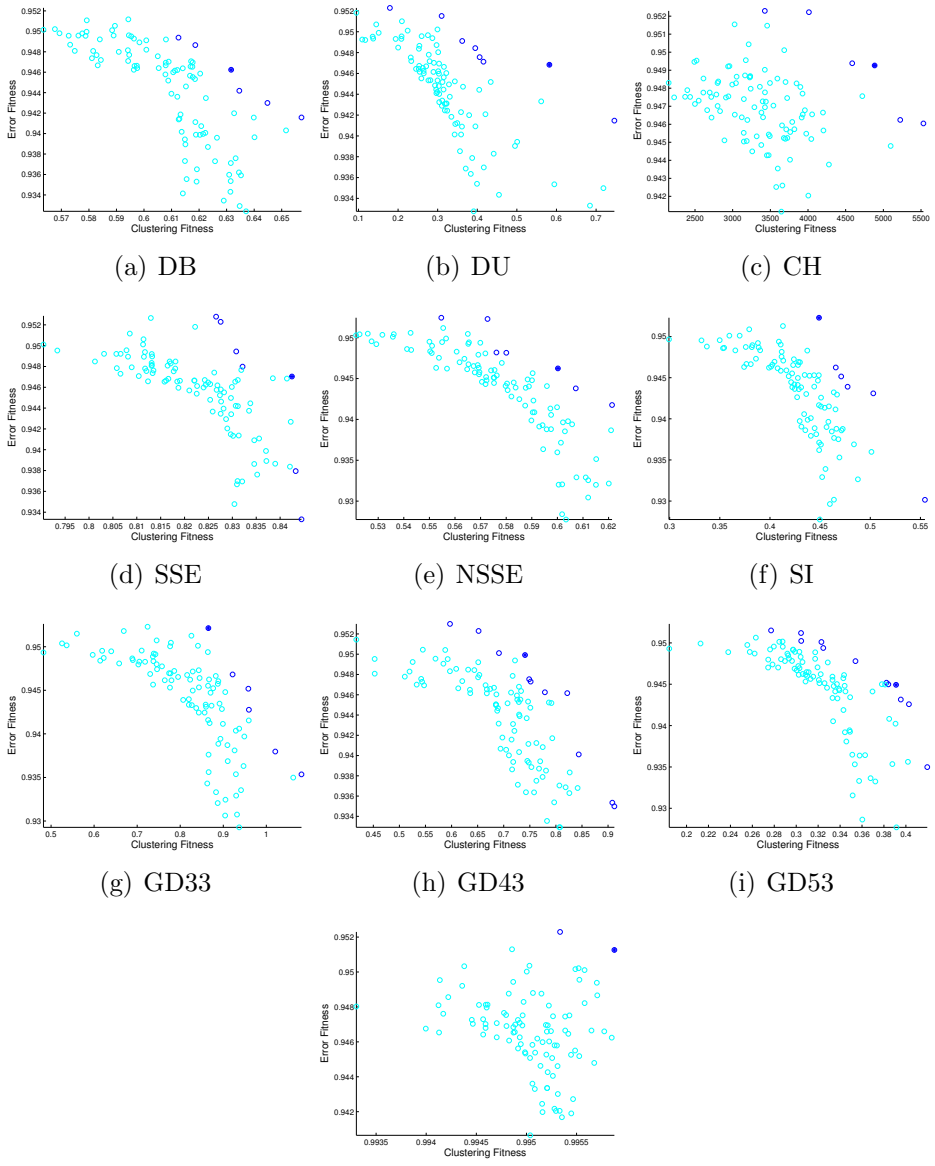
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.11: Frentes de Pareto para la serie *Arrhythmia*: RMSE vs *Clustering Fitness*.



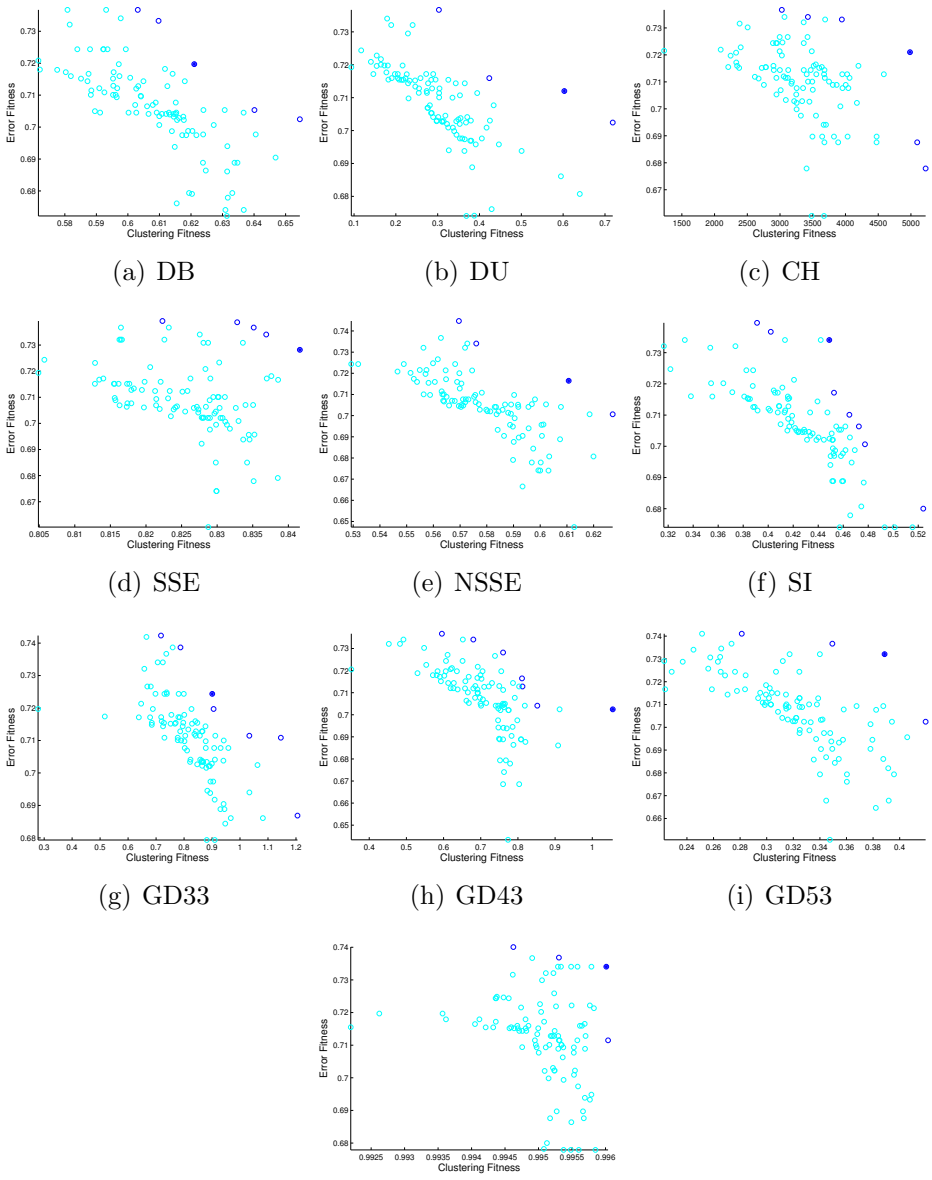
(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.12: Frentes de Pareto para la serie *Arrhythmia*: RMSEp vs *Clustering Fitness*.



(a) DB (b) DU (c) CH

(d) SSE (e) NSSE (f) SI

(g) GD33 (h) GD43 (i) GD53

(j) COP

Figura 4.13: Frentes de Pareto para la serie *Arrhythmia*: MAXe vs *Clustering Fitness*.

4.3. Experimento 2

En este experimento, se va a lanzar el algoritmo para cada serie temporal, con el fin de obtener unos resultados cuantitativos comparables con otros algoritmos del Estado del Arte, explicados en el capítulo 2, y así, demostrar que el algoritmo propuesto en términos globales es mejor que cualquier algoritmo mono-objetivo.

Para este propósito, y para cada serie, el algoritmo será ejecutado 30 veces con distintas semillas, y los resultados comparados serán la media de esas 30 ejecuciones, mostrando las soluciones extremas (las soluciones del frente de Pareto que dan mejores resultados en *clustering* o en error) y la mejor solución global. Esta última es aquella que en el frente de Pareto tiene una menor distancia Euclídea al óptimo (1,1).

Posteriormente se mostrará el resultado gráfico de la mejor ejecución en términos de error y de agrupamiento, para observar cualitativamente la aproximación de la serie y el agrupamiento realizados. En este caso, no se podrá mostrar la mejor solución global ya que, en una misma ejecución, hemos considerado la mejor solución global cómo aquella que se acerca más al punto (1,1) que sería el máximo global para ambas funciones de *fitness*. El problema es que ese punto se calcula mediante el escalado de los valores de *fitness* de la población, por tanto, ésta varía de una ejecución a otra.

4.3.1. Configuración de los Parámetros

Para este experimento la configuración de los parámetros ha sido:

- El tamaño mínimo de segmento es $min_{size} = 4$.
- El tamaño máximo de segmento es $max_{size} = 26{,}2\,\% \cdot longitud(serie)$, siendo $longitud(serie)$ la longitud de cada serie.
- El grado del polinomio es $n = 2$.

- El número de generaciones es $g = 200$.
- El tamaño de la población es $P = 200$.
- La probabilidad de cruce es $p_c = 0{,}8$.
- La probabilidad de mutación es $p_m = 0{,}2$.
- El porcentaje de puntos a ser mutados es del 20 %.
- El número de *clusters* se ha establecido en $k = 5$, en el caso de la serie del Ibex y $k = 4$ en el resto.
- El número de iteraciones para el *k-means* es $it = 20$.

4.3.2. Resultados y Discusión

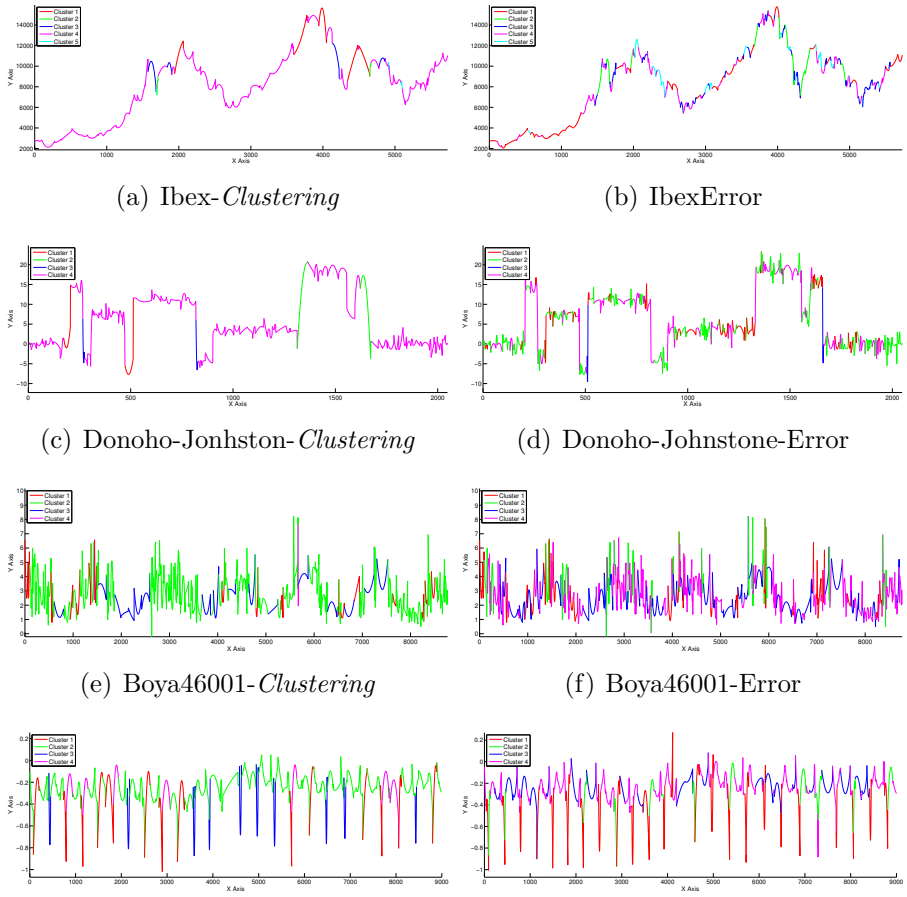
En la Tabla 4.1, se muestran los resultados del algoritmo con las series, mientras que en la Figura 4.14 se observan los resultados gráficos de la mejor solución en términos de agrupamiento y de la mejor solución en términos de error.

En todas las bases de datos ocurre el mismo fenómeno: cuando la mejor solución conlleva un *fitness* de *clustering* muy grande, los errores cometidos en la aproximación son mayores. Sin embargo, cuando minimizamos el error de aproximación, el índice de agrupamiento nos ofrece un valor peor. La mejor solución global, encuentra un equilibrio entre ambas medidas, ya que, aunque obviamente no alcanza los óptimos de los extremos, no comete un error tan grande como la medida contraria al óptimo de cada extremo.

El número de segmentos de cada solución también es importante: esta claro que si la solución posee un gran número de segmentos, la aproximación será mejor ya que se reduce el error cometido. Sin embargo, esto hace que el agrupamiento consiga detectar segmentos menos similares. De forma inversa, se produce el mismo problema.

# Segmentos	Dunn	RMSE	RMSEp	MAXe
		Ibex		
99.13 ± 18.07	1.32 ± 0.15	181.06 ± 18.57	194.60 ± 26.63	1348.78 ± 226.74
135.77 ± 11.58	0.41 ± 0.17	153.06 ± 5.15	156.44 ± 4.75	1105.65 ± 108.06
122.43 ± 12.60	1.03 ± 0.15	160.23 ± 6.95	164.12 ± 7.61	1174.10 ± 111.408
		Donoho-Johnstone		
120.17 ± 20.47	4.57 ± 1.00	2.10 ± 0.09	2.24 ± 0.09	11.83 ± 1.54
145.93 ± 17.95	0.57 ± 0.47	2.00 ± 0.05	2.10 ± 0.03	8.85 ± 0.76
141.07 ± 18.06	3.35 ± 0.80	2.02 ± 0.07	2.14 ± 0.04	10.20 ± 1.64
		Buoy 46001		
114.200 ± 24.13	1.25 ± 0.17	0.91 ± 0.04	0.96 ± 0.05	5.80 ± 0.26
170.57 ± 14.99	0.34 ± 0.11	0.83 ± 0.02	0.88 ± 0.02	5.48 ± 0.27
148.40± 20.68	0.93 ± 0.18	0.86 ± 0.03	0.91 ± 0.03	5.60 ± 0.34
		Arrhythmia		
104.97 ± 10.07	2.73 ± 0.32	0.06 ± 0.01	0.07 ± 0.01	0.53 ± 0.07
133.17 ± 8.62	0.67 ± 0.49	0.05 ± 0.00	0.05 ± 0.00	0.37 ± 0.04
121.10 ± 9.26	2.18 ± 0.26	0.05 ± 0.00	0.06 ± 0.00	0.42 ± 0.06

Tabla 4.1: Resultados del segundo experimento: Se muestran los resultados de la media de las 30 ejecuciones de la mejor solución en términos de agrupamiento (amarillo), en términos de error (naranja), y en términos globales (verde).



(a) Ibex-*Clustering* (b) IbexError

(c) Donoho-Jonhston-*Clustering* (d) Donoho-Johnstone-Error

(e) Boya46001-*Clustering* (f) Boya46001-Error

(g) Arrhythmia-*Clustering* (h) Arrhythmia-Error

Figura 4.14: Mejor solución de las 30 ejecuciones en agrupamiento y error.

4.4. Experimento 3

En este último experimento, se van a lanzar los algoritmos del Estado del Arte, para poder comparar los resultados obtenidos en el Experimento 4.3. Para ello, se va a ejecutar el algoritmo mono-objetivo que sólo tiene en cuenta el *fitness* de *clustering*, descrito en la sección 2.3. Este algoritmo ha sido adaptado a la codificación binaria y con el índice Dunn. Además, será configurado con los mismos parámetros que en el Experimento de la sección 4.3.

Por otro lado, se ejecutarán los algoritmos contenidos en la sección 2.4, exceptuando el *Top-Down* por su alto coste computacional. Estos algoritmos serán lanzados de forma que los resultados obtengan el mismo número de segmentos para cada una de las soluciones (mejor error, mejor agrupamiento y mejor global). Así pues, en esta sección no describiremos la configuración de los parámetros.

4.4.1. Resultados y Discusión

En la Tabla 4.2 se muestran los resultados obtenidos por los diferentes algoritmos ejecutados sobre la serie temporal del Ibex. Si atendemos a la mejor solución en términos de agrupamiento, podemos observar como el algoritmo evolutivo mono-objetivo basado en *clustering* consigue una mejor solución, obviamente porque sólo es optimizado para tal fin. Sin embargo, los resultados obtenidos en error por la aproximación son muy malos comparados con el algoritmo multiobjetivo. Los algoritmos de minimización del error de aproximación obtienen resultados muy parecidos al multiobjetivo (levemente mejores, especialmente en el *Bottom-Up*). En cuanto a la mejor solución en error encontrada por el algoritmo multiobjetivo, podemos destacar que obviamente el *clustering* es el peor de todos, pero en contrapartida los errores obtenidos son muy buenos, incluso superando a los algoritmos *Sliding Windows* y *SWAB*. Por último, cabe destacar que la mejor solución en términos globales, es decir, más cercana al óptimo (1,1), obtiene también resultados equiparables en error con los algoritmos

del Estado del Arte. Cabe destacar, que el máximo error encontrado por el multiobjetivo es muy grande en todos los experimentos.

En la Tabla 4.3 se muestran los resultados obtenidos por todos los algoritmos ejecutados sobre la base de datos temporal sintética *Donoho-Johnstone*. Al igual que en el caso anterior, la solución encontrada por el algoritmo evolutivo mono-objetivo es mejor en términos de agrupamiento que cualquier solución del algoritmo multiobjetivo. Sin embargo, en este caso, los algoritmos basados en el error de aproximación obtienen, en general, mejores resultados, en términos de error, que el algoritmo multiobjetivo evolutivo. No obstante los resultados son equiparables, y además, en el agrupamiento son mucho peores.

En las Tablas 4.4 y 4.5, aparecen los resultados obtenidos para las series restantes. Se observa el mismo resultado que en los casos anteriores, es decir, en *clustering* los resultados mejores son los obtenidos por el algoritmo mono-objetivo, y en error, los resultados de los algoritmos del Estado del Arte. Sin embargo, el multiobjetivo obtiene resultados comparables en agrupamiento con el mono-objetivo y muchos mejores en error; y comparables en error con los algoritmos de minimización del error de aproximación, pero mucho mejores en agrupamiento.

Algoritmo	# Segmentos	Dunn	RMSE	RMSEp	MAXe
Mejor solución en agrupamiento					
GMOTSS	99.13 ± 18.07	1.32 ± 0.15	181.06 ± 18.57	194.60 ± 26.63	1348.78 ± 226.74
EvolTSS	69.93 ± 6.82	1.60 ± 0.30	231.13 ± 12.77	274.03 ± 21.61	1799.33 ± 238.59
Sliding Window	99.00	0.18	177.33	189.48	770.34
Bottom up	100.00	0.40	150.74	167.37	770.50
SWAB	99.00	0.17	177.61	188.06	788.76
Mejor solución en error					
GMOTSS	135.77 ± 11.58	0.41 ± 0.17	153.06 ± 5.15	156.44 ± 4.75	1105.65 ± 108.06
EvolTSS	69.93 ± 6.82	1.60 ± 0.30	231.13 ± 12.77	274.03 ± 21.61	1799.33 ± 238.59
Sliding Window	136.00	0.13	152.10	164.24	709.75
Bottom up	136.00	0.29	129.72	147.27	548.68
SWAB	134.00	0.17	153.59	163.65	617.23
Mejor solución global					
GMOTSS	122.43 ± 12.60	1.03 ± 0.15	160.23 ± 6.95	164.12 ± 7.61	1174.10 ± 111.408
EvolTSS	69.93 ± 6.82	1.60 ± 0.30	231.13 ± 12.77	274.03 ± 21.61	1799.33 ± 238.59
Sliding Window	121.00	0.18	161.36	172.78	832.02
Bottom up	122.00	0.00	138.59	154.78	548.68
SWAB	123.00	0.18	160.89	170.44	832.02

Tabla 4.2: Resultados del Experimento 3 para la serie temporal: Ibex35.

Algoritmo	# Segmentos	Dunn	RMSE	RMSEp	MAXe
Mejor solución en agrupamiento					
GMOTSS	120.17 ± 20.47	4.57 ± 1.00	2.10 ± 0.09	2.24 ± 0.09	11.83 ± 1.54
EvolTSS	68.43 ± 6.91	5.46 ± 1.35	2.38 ± 0.08	2.59 ± 0.10	14.29 ± 1.83
Sliding Window	120.00	0.24	1.56	2.00	7.29
Bottom up	120.00	0.13	1.41	1.97	6.86
SWAB	120.00	0.24	1.56	1.99	7.29
Mejor solución en error					
GMOTSS	145.93 ± 17.95	0.57 ± 0.47	2.00 ± 0.05	2.10 ± 0.03	8.85 ± 0.76
EvolTSS	68.43 ± 6.91	5.46 ± 1.35	2.38 ± 0.08	2.59 ± 0.10	14.29 ± 1.83
Sliding Window	145.00	0.16	1.53	1.93	6.60
Bottom up	145.00	0.26	1.36	1.89	6.73
SWAB	145.00	0.16	1.53	1.94	6.60
Mejor solución global					
GMOTSS	141.07 ± 18.06	3.35 ± 0.80	2.02 ± 0.07	2.14 ± 0.04	10.20 ± 1.64
EvolTSS	68.43 ± 6.91	5.46 ± 1.35	2.38 ± 0.08	2.59 ± 0.10	14.29 ± 1.83
Sliding Window	141.00	0.28	1.51	1.93	6.32
Bottom up	141.00	0.28	1.37	1.90	6.73
SWAB	141.00	0.16	1.53	1.94	6.60

Tabla 4.3: Resultados del Experimento 3 para la serie temporal: Donoho-Johnstone.

Algoritmo	# Segmentos	Dunn	RMSE	RMSEp	MAXe
Mejor solución en agrupamiento					
GMOTSS	114.200 ± 24.13	1.25 ± 0.17	0.91 ± 0.04	0.96 ± 0.05	5.80 ± 0.26
EvolTSS	66.53 ± 10.21	1.67 ± 0.26	0.97 ± 0.04	1.07 ± 0.02	6.25 ± 0.27
Sliding Window	114.00	0.00	0.88	0.93	7.01
Bottom up	114.00	0.23	0.72	0.90	6.35
SWAB	114.00	0.23	0.72	0.89	6.36
Mejor solución en error					
GMOTSS	170.57 ± 14.99	0.34 ± 0.11	0.83 ± 0.02	0.88 ± 0.02	5.48 ± 0.27
EvolTSS	66.53 ± 10.21	1.67 ± 0.26	0.97 ± 0.04	1.07 ± 0.02	6.25 ± 0.27
Sliding Window	170.00	0.00	0.79	0.84	6.63
Bottom up	170.00	0.32	0.65	0.81	5.07
SWAB	171.00	0.13	0.65	0.81	5.07
Mejor solución global					
GMOTSS	148.40± 20.68	0.93 ± 0.18	0.86 ± 0.03	0.91 ± 0.03	5.60 ± 0.34
EvolTSS	66.53 ± 10.21	1.67 ± 0.26	0.97 ± 0.04	1.07 ± 0.02	6.25 ± 0.27
Sliding Window	148.00	0.00	0.83	0.88	6.70
Bottom up	148.00	0.28	0.67	0.85	6.40
SWAB	148.00	0.28	0.67	0.85	6.39

Tabla 4.4: Resultados del Experimento 3 para la serie temporal: Boya 46001.

Algoritmo	# Segmentos	Dunn	RMSE	RMSEp	MAXe
Mejor solución en agrupamiento					
GMOTSS	104.97 ± 10.07	2.73 ± 0.32	0.06 ± 0.01	0.07 ± 0.01	0.53 ± 0.07
EvolTSS	73.97 ± 6.53	3.82 ± 0.53	0.09 ± 0.01	0.11 ± 0.01	0.68 ± 0.03
Sliding Window	104.00	0.00	0.05	0.05	0.28
Bottom up	104.00	0.43	0.04	0.04	0.33
SWAB	104.00	0.43	0.04	0.04	0.33
Mejor solución en error					
GMOTSS	133.17 ± 8.62	0.67 ± 0.49	0.05 ± 0.00	0.05 ± 0.00	0.37 ± 0.04
EvolTSS	73.97 ± 6.53	3.82 ± 0.53	0.09 ± 0.01	0.11 ± 0.01	0.68 ± 0.03
Sliding Window	132.00	0.32	0.04	0.04	0.18
Bottom up	133.00	0.40	0.03	0.03	0.26
SWAB	133.00	0.40	0.03	0.03	0.26
Mejor solución global					
GMOTSS	121.10 ± 9.26	2.18 ± 0.26	0.05 ± 0.00	0.06 ± 0.00	0.42 ± 0.06
EvolTSS	73.97 ± 6.53	3.82 ± 0.53	0.09 ± 0.01	0.11 ± 0.01	0.68 ± 0.03
Sliding Window	121.00	0.52	0.04	0.04	0.18
Bottom up	120.00	0.00	0.03	0.04	0.32
SWAB	119.00	0.00	0.03	0.04	0.32

Tabla 4.5: Resultados del Experimento 3 para la serie temporal: *Arrhythmia*.

5 Conclusiones

Las conclusiones alcanzadas en el presente Trabajo Fin de Máster son las siguientes:

- Se ha realizado un estudio de los algoritmos de segmentación de series temporales existentes, de forma que se han implementado para adaptarlos a las codificaciones de este Trabajo Fin de Máster, y mostrar sus puntos débiles.
- Se ha realizado un estudio de los algoritmos multiobjetivo evolutivos y de los conceptos más importantes de éstos.
- Se ha implementado un algoritmo en el que sólo se tiene en cuenta el objetivo de aproximación de series temporales minimizando el error, mejorando los métodos ya existentes en este campo.
- Se han adaptado los métodos del algoritmo propuesto en Nikolaou et al. [2014], para el correcto funcionamiento de éstos en el presente Trabajo.
- Se ha implementado el primer algoritmo multiobjetivo evolutivo con la filosofía de disminuir el error cometido en la aproximación de series temporales mediante algoritmos de segmentación, además de aumentar la bondad del descubrimiento de patrones

similares con dicha aproximación. Demostrando que ambos objetivo son contrapuestos.

- Se ha ejecutado el algoritmo con base de datos de distinta naturaleza consiguiendo los resultados previstos y corroborando las hipótesis de partida.

- Se ha demostrado que una adecuada métrica de minimización del error global es el RMSEp.

- A medida que aumenta el número de divisiones en la segmentación, el valor del *clustering* disminuye mientras que el error de aproximación mejora.

- Es importante señalar que el presente trabajo ha servido para la realización de un artículo para un congreso internacional, que se adjunta en el Apéndice A del presente documento. Además, se está trabajando en otro artículo de revista que resume la metodología propuesta en este Trabajo Fin de Máster, así como una ampliación de los resultados obtenidos.

Gran cantidad de algoritmos son propuestos cada año, lo que causará que aparezcan nuevas mejoras. Por tanto, en cuanto a las futuras líneas de investigación abiertas a raíz del presente Trabajo Fin de Máster figuran, entre otras, las siguientes:

- Añadir una hibridación al algoritmo multiobjetivo evolutivo con un algoritmo de búsqueda local, de forma que se consiga una mejor explotación de soluciones, y en consecuencia soluciones más óptimas.

- Añadir nuevos operadores de cruce (por ejemplo, cruce multipadre).

- Añadir nuevos operadores de mutación, por ejemplo, la rotación cíclica.

- Hacer que el tamaño de las segmentaciones sea invariante de la evolución, mediante la modificación de los dos operadores anteriores.
- Adaptación del algoritmo para poder realizar predicción de series temporales. Para ello, se están estudiando las metodologías de previsión de segmentos de series temporales mediante modelos ocultos de *Markov*, y mediante redes de creencia profunda bayesiana.
- Adaptación del algoritmo para la correcta segmentación de series temporales para la detección de rampas de viento, y para la predicción de energía eólica.

Bibliografía

O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Perez, and I. n. Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243 – 256, 2013.

D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-24-5.

S. Bandyopadhyay and S. Saha. A point symmetry-based clustering technique for automatic evolution of clusters. *Knowledge and Data Engineering, IEEE Transactions on*, 20(11):1441–1457, Nov 2008. ISSN 1041-4347.

J. Chen. Making subsequence time series clustering meaningful. *The IEEE International Conference on Data Mining*, pages 114–121, 2005.

F.-L. Chung, T.-C. Fu, V. Ng, and R. W. Luk. An evolutionary approach to pattern-based time series segmentation. *Evolutionary Computation, IEEE Transactions on*, 8(5):471–489, 2004.

G. Das, K. ip Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. pages 16–22. AAAI Press, 1998.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Compu-*

tation, IEEE Transactions on, 6(2):182–197, Apr 2002. ISSN 1089-778X. doi: 10.1109/4235.996017.

D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90:1200–1224, 1995.

D. L. Donoho and J. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994. doi: 10.1093/biomet/81.3.425.

D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: Asymptopia? *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(2):301–369, 1995. ISSN 00359246.

E. Fuchs, T. Gruber, J. Nitschke, and B. Sick. On-line motif detection in time series with swiftmotif. *Pattern Recognition*, 42 (11):3015 – 3031, 2009. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2009.05.004. URL `http://www.sciencedirect.com/science/article/pii/S003132030900171X`.

A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000. doi: 10.1161/01.CIR.101.23.e215.

E. Keogh, J. Lin, and W. Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future. *The IEEE International Conference on Data Mining*, pages 115–122, 2003.

E. J. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting Time Series: A Survey and Novel Approach. In M. Last, A. Kandel, and H. Bunke, editors, *Data Mining In Time Series Databases*, volume 57 of *Series in Machine Perception and Artificial Intelligence*, chapter 1, pages 1–22. World Scientific Publishing Company, 2004. ISBN 978-981-238-290-0.

G. Moody and R. Mark. The impact of the mit-bih arrhythmia database. *Engineering in Medicine and Biology Magazine, IEEE*, 20(3): 45–50, May 2001. ISSN 0739-5175. doi: 10.1109/51.932724.

NDBC. National buoy data center. National Oceanic and Atmospheric Administration of the USA (NOAA), 2015. URL `http://www.ndbc.noaa.gov/`.

A. Nikolaou, P. Gutiérrez, A. Durán, I. Dicaire, F. Fernández-Navarro, and C. Hervás-Martínez. Detection of early warning signals in paleoclimate data using a genetic time series segmentation algorithm. *Climate Dynamics*, pages 1–15, 2014. ISSN 0930-7575. doi: 10.1007/s00382-014-2405-0. URL `http://dx.doi.org/10.1007/s00382-014-2405-0`.

J. Oliver and C. Forbes. Bayesian approaches to segmenting a simple time series. Technical Report 14/97, Monash University, Department of Econometrics and Business Statistics, 1997. URL `http://EconPapers.repec.org/RePEc:msh:ebswps:1997-14`.

J. J. Oliver, R. A. Baxter, and C. S. Wallace. Minimum message length segmentation. In X. Wu, R. Kotagiri, and K. Korb, editors, *Research and Development in Knowledge Discovery and Data Mining*, volume 1394 of *Lecture Notes in Computer Science*, pages 222–233. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-64383-8. doi: 10.1007/3-540-64383-4_19. URL `http://dx.doi.org/10.1007/3-540-64383-4_19`.

D. Percival and A. Walden. Wavelet methods for time serie analysis. *Cambridge University Press*, 2000.

V. S. Tseng, C.-H. Chen, P.-C. Huang, and T.-P. Hong. Cluster-based genetic segmentation of time series with dwt. *Pattern Recognition Letters*, 30:1190–1197, 2009.

R. Xu and D. Wunsch. *Clustering*. IEEE Press Series on Computational Intelligence. Wiley, 2008. ISBN 9780470382783.

Parte II

Apéndices

A Primer apéndice: Artículo adjunto

Se adjunta como primer apéndice al documento, un artículo realizado por el autor y los directores del presente Trabajo Fin de Máster. **Actualmente, el artículo está aceptado para publicación en el Congreso Internacional *Hybrid Artificial Intelligence Systems, HAIS2016.***

El artículo aplica una primera versión del algoritmo, en el que sólo se tiene en cuenta el objetivo de minimización del error de aproximación en series temporales mediante un algoritmo de segmentación. Se aplica sobre dos de las cuatro bases de datos presentadas en este documento, concretamente, la serie temporal del IBEX35 y la de Donoho-Jonhstone.

Time series representation by a novel hybrid segmentation algorithm*

A.M. Durán-Rosal[1]**, P.A. Gutiérrez[1], F.J. Martínez-Estudillo[2], and C. Hervás-Martínez[1]

[1] University of Córdoba, Dept. of Computer Science and Numerical Analysis, Rabanales Campus, Albert Einstein building, 14071 - Córdoba, Spain
{i92duroa,pagutierrez,chervas}@uco.es
[2] Department of Quantitative Methods, Loyola Andalucía University, Escritor Castilla Aguayo 4, 14004 Córdoba, Spain {fjmestud}@uloyola.es

Abstract. Time series representation can be approached by segmentation genetic algorithms (GAs) with the purpose of automatically finding segments approximating the time series with the lowest possible error. Although this is an interesting data mining field, obtaining the optimal segmentation of time series in different scopes is a very challenging task. In this way, very accurate algorithms are needed. On the other hand, it is well-known that GAs are relatively poor when finding the precise optimum solution in the region where they converge. Thus, this paper presents a hybrid GA algorithm including a local search method, aimed to improve the quality of the final solution. The local search algorithm is based on two well-known algorithms: Bottom-Up and Top-Down. A real-world time series in the Spanish Stock Market field (IBEX35) and a synthetic database (Donoho-Johnstone) used in other researches were used to test the proposed methodology.

Keywords: Time series segmentation, hybrid algorithms, time series representation, Spanish Stock Market Index, synthetic database

1 Introduction

Recently, the ubiquity of temporal data has initiated significant research and development efforts in the field of data mining. In this sense, time series can be easily obtained from financial and scientific applications, being one of the main sources of temporal datasets. Data mining in time series databases involves several tasks. One important task is the segmentation of time series, which consists

* This work has been subsidized by the project TIN2014-54583-C2-1-R of the Spanish Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P11-TIC-7508 project of the Junta de Andalucía (Spain). Antonio M. Durán-Rosal's research has been subsidized by the FPU Predoctoral Program (Spanish Ministry of Education and Science), grant reference FPU14/03039.

** Corresponding author at: Tel.: +34 957 218 349; Fax: +34 957 218 630.

in cutting the time series in some specific points, trying to achieve different objectives. The two main objectives in time series segmentation are now described.

One the one hand, discovering useful patterns in the time series is an appealing objective [1]. Das *et al.* [2] used a fixed-length window to segment the time series and represent it using the resulting simple patterns. Related with this, the characterization of strange events or special patterns in a time series (which are known as "tipping points") [3] can be used to discover common patterns which occur before these events and to extract conclusions about their nature. Finally, finding segments is also a way to discover temporal patterns or anomalies [4].

On the other hand, simplifying the time series by a segmentation algorithm is an option to alleviate the difficulty of processing, analysing or mining time series databases, given their continuous nature. Previous approaches [5,6] suggest dividing the time series using previously identified change points and substituting the segments with suitable approximations. We focus on a simplification based on piecewise linear approximation (PLA), which is able to reduce the number of points in the time series with a minimum reduction of information [7].

Genetic algorithms (GAs) are able to perform a global multi-point search, converging to high quality areas. For this reason, they are considered robust heuristics. However, they are not good at finding the precise optimum solution in these areas [8]. To solve this problem, during the last years, local optimization has been incorporated to GAs to refine their results. The idea is to combine genetic algorithms, as global explorers, with these local search (LS) procedures, which are good at finding local optima (local exploiters), resulting in what is usually known as hybrid algorithms.

There are several ways in which a LS procedure can be combined with an EAs. The way the combination is done is extremely important in terms of accuracy and computational efficiency, and the best balance of local exploitation and global exploration has to be obtained. Some of the strategies previously used include the multistart approach, the Lamarckian learning, the Baldwinian learning, the partial Lamarckianism and the process of random linkage [9,10,11].

In this paper, we propose a hybrid genetic algorithm for segmenting time-series and reducing their dimensionality using the PLA. The methodology combines a GA (global explorer) and a new LS procedure based on the well-known Bottom-Up and Top-Down methods [7]. The LS process consists in removing a number of cut points by the Bottom-Up methodology and, then, add the same number of points using the Top-Down procedure.

The LS is applied to the best solution obtained by the GA in the last generation, to find the precise local optimum around the final solution thanks to the high quality area returned by the GA. The results of the algorithm have been obtained with and without the LS process, and compared to the original Bottom-Up and Top-Down procedures. To test the performance of the proposed hybrid method, it is applied to a hard real-world time series in the Stock Market field (IBEX 35) and to the synthetic database Donoho-Jonhstone.

The rest of the paper is organized as follows. Section 2 includes the characteristics of the proposed algorithm, while Section 3 presents the description of

the time series, the experiments performed and the discussion about the results. Finally, Section 4 establishes the conclusions.

2 Hybrid segmentation algorithm

2.1 Summary of the algorithm

Given a time series $Y = \{y_n\}_{n=1}^{N}$, our objective is to divide the values of y_n into m consecutive subsets or segments, where the error of the approximation of these m segments should be as less as possible. Note that the search space is wide, its size being $\binom{N}{m}$.

The time indexes $(n = 1, \ldots, N)$ are divided into segments: $s_1 = \{y_1, \ldots, y_{t_1}\}$, $s_2 = \{y_{t_1}, \ldots, y_{t_2}\}, \ldots, s_m = \{y_{t_{m-1}}, \ldots, y_N\}$, where the ts are the different cut points subscripted in ascending order $(t_1 < t_2 < \ldots < t_{m-1})$. The cut points are the only points which belong to two segments (the segment before and the segment after). The values of the cut points, $t_i, i = 1, \ldots, m-1$, have to be determined by the algorithm. In this work, we have approximated each segment using a linear interpolation between the initial and final cut points of the segment. The main steps of the algorithm are summarized in Fig. 1.

2.2 Genetic Algorithm

The characteristics of the GA are defined as follows.

Chromosome representation Each individual chromosome (c) consists of an array of binary values, where the length of the chromosome is the time series length, N. Each position c_{t_i} stores whether the time index t_i of the time series represents a cut point for the evaluated solution. In this way, $\sum_{i=1}^{N} c_{t_i} = m$.

Initial population The population of the GA is a set P of binary vectors of length N, where m positions are $1s$ (the cut points, which are randomly chosen with a uniform distribution) and the rest are $0s$.

Fitness evaluation The goal of this kind of segmentation procedure is to reduce the error between the complete time series and its approximation. The fitness function could be defined as minimizing the difference between each real value of the time series and the corresponding approximation. The error of the i-th point in the chromosome c is defined as $e_i(c) = y_i - \bar{y}_i$, where y_i is the real value, and $\bar{y}_i(c)$ is the approximation of the chromosome c. We have considered two different fitness functions, which lead to very different results:

– The first function is the Root Mean Squared Error (RMSE), and it is calculated as follows:

$$RMSE(c) = \sqrt{MSE_c} = \sqrt{\frac{1}{m}\sum_{s=1}^{m} MSE_{c,s}} = \\ = \sqrt{\frac{1}{m}\sum_{s=1}^{m} \frac{1}{t_s - t_{s-1}+1}\sum_{i=t_{s-1}}^{t_s} e_i^2(c)}$$

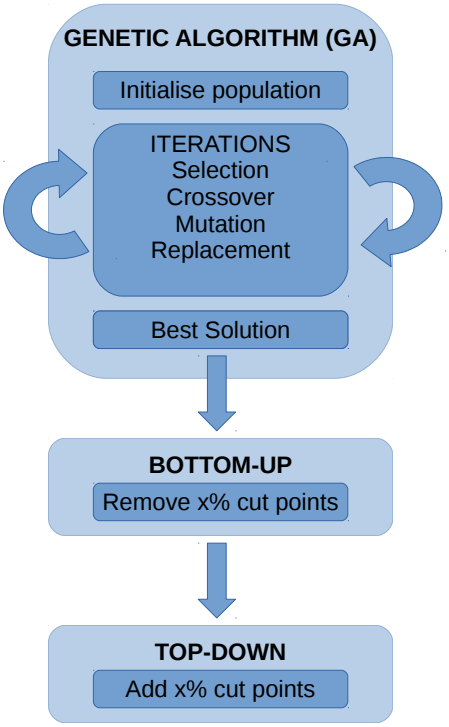


Fig. 1. Main steps for the proposed hybrid algorithm

where $MSE_s(c)$ is the Mean Squared Error (MSE) of each segment s, and MSE_c is the average value of the $MSE_{c,s}$ for all segments.

– The second function (called RMSEp, the error being averaged at a point-level), is obtained as follows:

$$RMSEp(c) = \sqrt{\frac{1}{N} SSE(c)} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} e_i^2(c)}$$

where, as can be seen, the Sum of Square Errors ($SSE(c)$) is used, which does not average the errors of the points for each segment.

As both metrics have to be minimised, the fitness function is defined as $f = \frac{1}{1+RMSE}$ and $f = \frac{1}{1+RMSEp}$, respectively, which are bounded in the interval $[0, 1]$.

Selection and replacement processes All the individuals are considered for reproduction and generation of offspring. However, a replacement process is

applied to the joint offspring and parent populations by roulette wheel selection. The selection process promotes diversity, but the replacement process is promoting selective pressure.

Mutation and Crossover Operators Two kinds of operators are included in the algorithm to reduce the dependency with respect to the initial population and escape from local optima:

- Mutation operator: the probability p_{m} of performing any mutation is decided by the user. The kind of mutation applied to the individual is randomly selected from the following two operators: cyclic rotation to the left or to the right, where the number of positions in the chromosome are randomly chosen in the interval $[1, N-1]$
- Crossover operator: For each parent individual (c^1), the crossover operator is applied with a given probability p_{c}. The operator randomly selects the other parent (c^2) and a time index (t_j). Then, those positions $t_i \geq t_j$ where $c^1_{t_i} = 1 \wedge c^2_{t_i} = 0$ are interchanged and also those where $c^1_{t_i} = 1 \wedge c^2_{t_i} = 0$. However, in order to maintain the same number of cutting points in the offspring, the maximum number of interchanges allowed is the minimum of the number of positions where the conditions are fulfilled.

2.3 Local search

When the genetic algorithm is finished, the best solution in the last generation, is improved by with two well-known segmentation algorithms: **Bottom-Up** and **Top-Down**. On the one hand, in each iteration, Bottom-Up consists in merging the two adjacent segments with the lowest cost of merging (lowest error after merging). On the other hand, Top-Down is the natural complement to the Bottom-Up algorithm, recursively partitioning the time series until some stopping criteria (related with the error) is met. The proposed local search method removes a percentage of cut points using the Bottom-up strategy and then adds the same number of cut point with the Top-Down technique.

To use these algorithms, we have modified the implementation proposed in [7], in such a way that, for both, the stopping criteria is the number of segments to merge or cut, respectively. Note that the implementation of Top-Down presented in [7] is recursive method, so we have transformed it into an iterative method to know how many points have been added up to a given iteration.

3 Experimental results and discussion

The experiments performed and the results obtained are analysed in this section.

3.1 Time series analysed

In this paper, we evaluate the performance of the algorithm in two time series from different areas, to test its adaptability in various problems. The series considered are the following ones:

- The first time series is the IBEX35. It is one of the official indexes of the Madrid Stock Market, which is composed of the most liquid values listed in the Computer Assisted Trading System. In our study, we have considered the daily closing prices of this index from 4th January 1992 to 26th September 2014, presenting thus a total of 5730 observations[3]. The complete time series can be seen in Fig. 2 (a).
- The second time series is extracted from a benchmark repository used by the neural net and machine learning community [12,13,14]. The datasets in this repository tend to have many inputs, either no noise of lots of noise, and little to moderate nonlinearity. The Donoho-Jonhstone benchmarks consist of four functions (called Blocks, Bumps, Heavisine, and Doppler) to which random noise can be added to produce an infinitive number of data sets. In our experiments, we used the function Blocks with medium noise, with a total of 2048 observations [4]. The complete time series is included in Fig. 3 (a).

As we can check, the time series have different length and comes from different sources.

3.2 Experimental setting

The experimental design for the time series under study is presented in this subsection.

Both fitness fuctions (RMSE and RMSEp) were considered in different experiments. The GA was configured with the following parameter values, obtained by a *trial and error* procedure. The number of individuals of the population is fixed to $P = 100$. The crossover probability is taken as $p_c = 0.8$ and the mutation probability as $p_c = 0.2$. The maximum number of generations is configured as $g = 200$. The number of points of the aproximation was established at $m = 2.5N\%$, where N is the length of the time series. The percentage of cut points to perform the LS procedure after the GA is set to 40%. Given the stochastic nature of GAs, the algorithm was run 30 times with different seeds, and we obtained the mean and the standard deviation of these results. When applying only LS procedures, one single result is recorded are these are deterministic methods.

3.3 Discussion

Tables 1, 2, 3 include, respectively, the results obtained using RMSE as fitness function (for GAs) or cutting/merging criterion (for LS procedures), those obtained using RMSEp and the correlation coefficient (r^2) of the approximated time series with respect to the original one. For this last table, in the case of GAs, the best fitness solution of the 30 ones has been selected.

[3] The corresponding values can be downloaded at `https://es.finance.yahoo.com/`
[4] The time series can be downloaded at `https://sites.google.com/site/icdmmdl/`

First of all, we compare the performance of the GA without LS (GTSS) to the performance obtained when hybridizing it with the LS procedure (HTTS). On the one hand, results in Table 1 show that, when considering RMSE as fitness function, we reduce to a great extent the RMSE if the LS procedure is applied after the GA. Furthermore, the dispersion of the results (standard deviation) is also lower, what indicates that the algorithm tends to provide similar solutions in terms of fitness. On the other hand, Table 2 shows that the same can be observed when using the RMSEp function. The improvement is even greater in this case, because it reduces the RMSEp in 76 points for the IBEX time series and 0.4 for Dohono-Johnstone. The dispersion is still relatively low.

Comparing the GAs with respect to the LS algorithms, the following observations can be made. Table 1 shows that the best approximation is obtained by the HTSS method in the case of IBEX and using the RMSE fitness function. Furthermore, GTSS obtains better RMSE than LS algorithms. In Donoho-Jonhstone time series, the best approximation in terms of RMSE is Top-Down. However, we should take into account that the RMSE function can be deceiving for this purpose because it tries to minimize the MSE of the different segments, independently of their length. In this way, Top-Down minimises the error for a part of the time series with a lot of very short segments, the RMSE counteracting the error of the long segments with those errors of the short ones. This problem can be observed in the columns of the RMSE fitness function of Table 3, where the approximation of IBEX with Top-Down is the worst in terms of r^2 using RMSE and RMSEp, and the approximation of Donoho-Jonhstone is also bad using HTSS and TOP-DOWN. We can observe graphically this problem in Fig. 3 (b) (f) and Fig. 2 (f). For this reason, RMSE function is not suitable for representing the global error of the time series approximation, and we recommend using RMSEp.

Results with RMSEp are more confident, because the best methods in Table 2 are HTSS and Bottom-Up, both methods showing a good graphical behaviour in Figures 2 and 3 and a good correlation coefficient in Table 3. It is important to mention that, using this metric, we reduce the problem of the RMSE for Top-Down, and we totally avoid it for the case of HTSS.

Finally, as said before, the best approximation in terms of r^2 are obtained with the HTSS algorithm. Although Bottom-Up obtains similar results in r^2, HTSS is better in error terms, so it is the recommended algorithm for future tasks.

4 Conclusions

This paper presents a novel hybrid time series segmentation method to simplify time series using PLAs, combining a GA with a LS. The LS sequentially applies Bottom-Up and Top-Down methods. We have tested the method in two time-series: the Spanish Stock market IBEX35 index and the synthetic Donoho-Johnstone database, showing that the results are better with the hybrid method than with LS algorithms. Experiments also reveals that the RMSE metric does

Algorithm	IBEX	Donoho-Johnstone
GTSS	*188.604 ± 11.933*	1.716 ± 0.226
HTSS	**160.185 ± 8.045**	*1.531 ± 0.193*
BOTTOM-UP	211.799	2.127
TOP-DOWN	206.729	**1.144**

Table 1. Results (Mean±StandardDeviation for the GAs or one single result for the LS methods) of the different algorithms using RMSE as the fitness function (for the GAs) or as the cutting/merging criterion (for the LS methods).

Algorithm	IBEX	Donoho-Johnstone
GTSS	280.905 ± 9.551	2.927 ± 0.080
HTSS	**204.195 ± 3.641**	**2.581 ± 0.054**
BOTTOM-UP	*210.321*	*2.639*
TOP-DOWN	269.801	3.466

Table 2. Results (Mean±StandardDeviation for the GAs or one single result for the LS methods) of the different algorithms using RMSEp as the fitness function (for the GAs) or as the cut/merge criterion (for the LS methods).

	IBEX		Donoho-Johnstone	
Fitness	RMSE - r^2	RMSEp - r^2	RMSE - r^2	RMSEp - r^2
HTSS	0.9932	**0.9982**	0.5560	**0.9407**
BOTTOM-UP	0.9973	*0.9980*	0.9275	*0.9351*
TOP-DOWN	0.8230	0.9967	0.1053	0.8810

Table 3. Correlation coefficient comparing the similarity of the approximated time series with the original one.

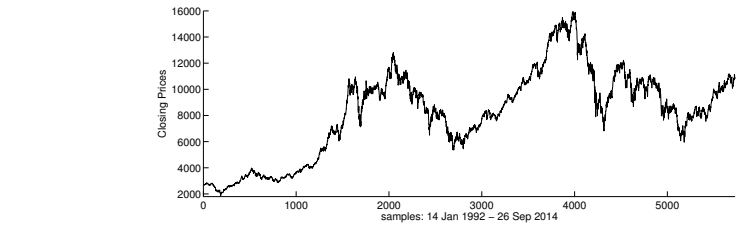
not lead to a good global approximation of the time series. In this sense, the RMSEp is a better option, where the error is averaged at a point level.

We plan to extend this work in several directions:

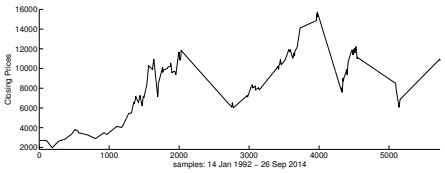
- To use more time series for validating the conclusions.
- To test the proposed method in other tasks, as classification or clustering, comparing the results in these tasks using the original time series and the approximated versions.
- To approximate the time series applying linear regression instead of linear interpolation, or even using polynomials with degree greater than one, not necessarily linear.

References

1. Chung, F.L., Fu, T.C., Ng, V., Luk, R.W.: An evolutionary approach to pattern-based time series segmentation. Evolutionary Computation, IEEE Transactions on **8**(5) (2004) 471–489
2. Das, G., ip Lin, K., Mannila, H., Renganathan, G., Smyth, P.: Rule discovery from time series, AAAI Press (1998) 16–22
3. Nikolaou, A., Gutiérrez, P., Durán, A., Dicaire, I., Fernández-Navarro, F., Hervás-Martínez, C.: Detection of early warning signals in paleoclimate data using a genetic time series segmentation algorithm. Climate Dynamics (2014) 1–15
4. Fuchs, E., Gruber, T., Nitschke, J., Sick, B.: On-line motif detection in time series with swiftmotif. Pattern Recognition **42**(11) (2009) 3015 – 3031
5. Oliver, J., Forbes, C.: Bayesian approaches to segmenting a simple time series. Technical Report 14/97, Monash University, Department of Econometrics and Business Statistics (1997)
6. Oliver, J.J., Baxter, R.A., Wallace, C.S.: Minimum message length segmentation. In Wu, X., Kotagiri, R., Korb, K., eds.: Research and Development in Knowledge Discovery and Data Mining. Volume 1394 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1998) 222–233
7. Keogh, E.J., Chu, S., Hart, D., Pazzani, M.: Segmenting Time Series: A Survey and Novel Approach. In Last, M., Kandel, A., Bunke, H., eds.: Data Mining In Time Series Databases. Volume 57 of Series in Machine Perception and Artificial Intelligence. World Scientific Publishing Company (2004) 1–22
8. Houck, C.R., Joines, J.A., Kay, M.G., Wilson, J.R.: Empirical investigation of the benefits of partial lamarckianism. Evol. Comput. **5**(1) (March 1997) 31–60
9. Kolen, A., Pesch, E.: Genetic local search in combinatorial optimization. Discrete Applied Mathematics **48**(3) (1994) 273 – 284
10. Joines, J.A., Kay, M.G.: Utilizing hybrid genetic algorithms. In: Evolutionary Optimization. Volume 48 of International Series in Operations Research & Management Science. Springer US (2002) 199–228
11. Ulder, N.L.J., Aarts, E.H.L., Bandelt, H.J., Laarhoven, P.J.M.v., Pesch, E.: Genetic local search algorithms for the travelling salesman problem. In: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature. PPSN I, London, UK, UK, Springer-Verlag (1991) 109–116
12. Donoho, D.L., Johnstone, J.M.: Ideal spatial adaptation by wavelet shrinkage. Biometrika **81**(3) (1994) 425–455
13. Donoho, D.L., Johnstone, I.M., Kerkyacharian, G., Picard, D.: Wavelet shrinkage: Asymptopia? Journal of the Royal Statistical Society. Series B (Methodological) **57**(2) (1995) 301–369
14. Donoho, D.L., Johnstone, I.M.: Adapting to unknown smoothness via wavelet shrinkage. Journal of the American Statistical Association **90** (1995) 1200–1224

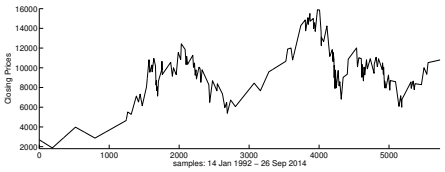


(a) Original Time Series



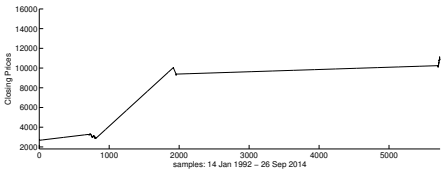
(b) HTSS using RMSE

(c) HTSS using RMSEp

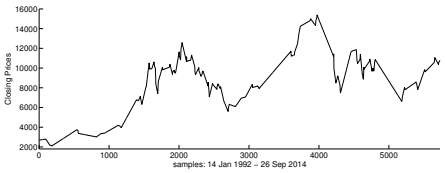


(d) Bottom-up using RMSE

(e) Bottom-up using RMSEp

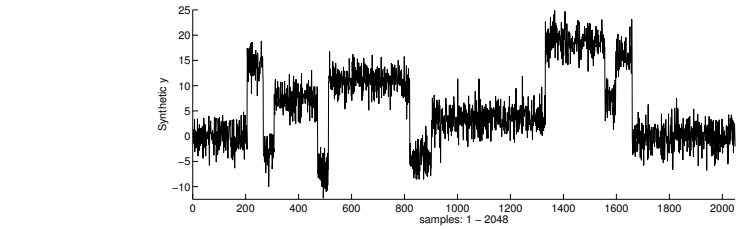


(f) Top-Down using RMSE

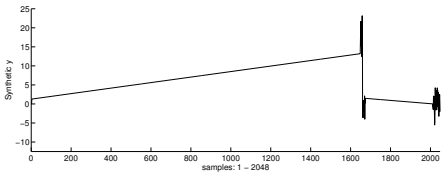


(g) Top-Down using RMSEp

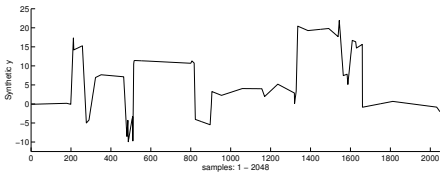
Fig. 2. Original IBEX time series and approximations by the different methods compared.



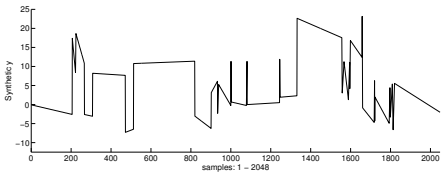
(a) Original Time Series



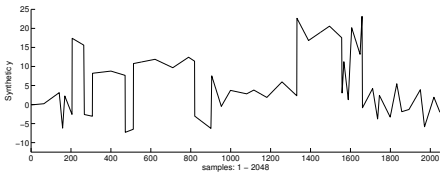
(b) HTSS using RMSE



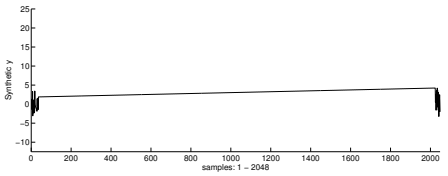
(c) HTSS using RMSEp



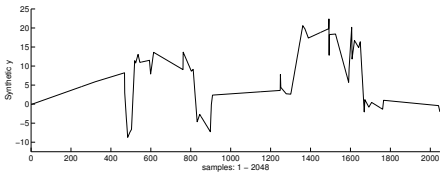
(d) Bottom-up using RMSE



(e) Bottom-up using RMSEp



(f) Top-Down using RMSE



(g) Top-Down using RMSEp

Fig. 3. Original Donoho-Jonhstone time series and approximations by the different methods compared.

B Segundo apéndice: Artículos

Se adjunta como segundo apéndice al documento, una lista de artículos que han sido realizados por el autor de este Trabajo Fin de Máster y que guardan relación con éste. A continuación, se mostrarán el conjunto de trabajos realizados con la primera página de cada uno.

- *"Detection of early warning signals in palecolimate data using a genetic time series segmentation algorithm"*. Artículo **publicado** en 2015, en la revista *Climate Dynamics*.
- *"Applying a Hybrid Algorithm to the Segmentation of the Stock Market Index Time Series"*. Artículo **publicado** en 2015, en *The International Work-Conference on Artificial Neural Networks (IWANN)*.
- *"Massive Missing Data Reconstruction in Ocean Buoys with Evolutionary Product Unit Neural Networks"*. Artículo en **segunda revisión** en la revista *Ocean Engineering*.
- *"On the use of evolutionary time series analysis for segmenting paleoclimate data"*. Artículo **enviado** a la revista *Neurocomputing*.

- *"Identifying market behaviours using European Stock Index time series by a hybrid segmentation algorithm"*. Artículo **enviado como ampliación** a la revista *Neural Processing Letters*.

Clim Dyn (2015) 44:1919–1933
DOI 10.1007/s00382-014-2405-0

Detection of early warning signals in paleoclimate data using a genetic time series segmentation algorithm

Athanasia Nikolaou · Pedro Antonio Gutiérrez · Antonio Durán · Isabelle Dicaire · Francisco Fernández-Navarro · César Hervás-Martínez

Received: 29 November 2013 / Accepted: 6 November 2014 / Published online: 26 November 2014

Abstract This paper proposes a time series segmentation algorithm combining a clustering technique and a genetic algorithm to automatically find segments sharing common statistical characteristics in paleoclimate time series. The segments are transformed into a six-dimensional space composed of six statistical measures, most of which have been previously considered in the detection of warning signals of critical transitions. Experimental results show that the proposed approach applied to paleoclimate data could effectively analyse Dansgaard–Oeschger (DO) events and uncover commonalities and differences in their statistical and possibly their dynamical characterisation. In particular, warning signals were robustly detected in the GISP2 and NGRIP δ^{18}O ice core data for several DO events (e.g. DO 1, 4, 8 and 12) in the form of an order of magnitude increase in variance, autocorrelation and mean square distance from a linear approximation (i.e. the mean square error). The increase in mean square error, suggesting nonlinear behaviour, has been found to correspond with an increase in variance prior to several DO events for ∼90 % of the algorithm runs for the GISP2 δ^{18}O dataset and for ∼100 % of the algorithm runs for the NGRIP δ^{18}O dataset. The proposed approach applied to well-known dynamical systems and paleoclimate datasets provides a novel visualisation tool in the field of climate time series analysis.

Keywords Warning signals · Time series segmentation · Tipping points · Abrupt climate change · Genetic algorithms · Clustering

A. Nikolaou · I. Dicaire (✉) · F. Fernández-Navarro
Advanced Concepts Team, European Space Research and Technology Centre (ESTEC), European Space Agency (ESA), Noordwijk, Netherlands
e-mail: isabelle.dicaire@esa.int

A. Nikolaou
e-mail: athanasia.nikolaou@esa.int

F. Fernández-Navarro
e-mail: i22fenaf@uco.es; fafernandez@uloyola.es

P. A. Gutiérrez · A. Durán · C. Hervás-Martínez
Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba, Spain
e-mail: pagutierrez@uco.es

A. Durán
e-mail: i92duroa@uco.es

C. Hervás-Martínez
e-mail: chervas@uco.es

F. Fernández-Navarro
Department of Mathematics and Engineering, Universidad Loyola Andalucia, Seville, Spain

1 Introduction

The statistical tools used to extract knowledge from time series analysis have undergone considerable development during the past decade (see Livina and Lenton 2007; Livina et al. 2011; Lenton et al. 2012; Scheffer et al. 2009; Dakos et al. 2008; Held and Kleinen 2004; Cimatoribus et al. 2013). Driven by the ultimate aim of understanding past climate variability, the above studies focused on statistical analysis of time series that demonstrate *threshold behaviour* as used in Alley et al. (2003). Candidate explanations for transitions of a system over thresholds link to dynamical systems analysis, which is used for gaining insight into internal variability modes and response to external forcing on both simple and complex systems (Saltzman 2001). Adopting the notation from Ashwin et al. (2012) the abrupt shift from a stable state to another stable state could be e.g. due to *B-tipping* or *N-tipping*. In B-tipping the system is driven past bifurcation points, where equilibrium solutions

Applying a hybrid algorithm to the segmentation of the Spanish stock market index time series*

A.M. Durán-Rosal[1]**, M. de la Paz-Marín[2], P.A. Gutiérrez[1], and C. Hervás-Martínez[1]

[1] University of Córdoba, Dept. of Computer Science and Numerical Analysis, Rabanales Campus, Albert Einstein building, 14071 - Córdoba, Spain
{i92duroa,pagutierrez,chervas}@uco.es
[2] Department of Management and Quantitative Methods, Loyola Andalucía University, Business Administration Faculty, Escritor Castilla Aguayo 4, 14004 Córdoba, Spain {mpaz}@uco.es

Abstract. Time-series segmentation can be approached by combining a clustering technique and genetic algorithm (GA) with the purpose of automatically finding segments and patterns of a time series. This is an interesting data mining field, but its application to the optimal segmentation of financial time series is a very challenging task, so accurate algorithms are needed. In this sense, GAs are relatively poor at finding the precise optimum solution in the region where the algorithm converges. Thus, this work presents a hybrid GA algorithm including a local search method, aimed to improve the quality of the final solution. The local search algorithm is based on maximizing a likelihood ratio, assuming normality for the series and the subseries in which the original one is segmented. A real-world time series in the Spanish Stock Market field was used to test this methodology.

Keywords: Time series segmentation, hybrid algorithms, clustering, Spanish Stock Market Index.

1 Introduction

Recently, the ubiquity of temporal data has initiated various research and development efforts in the field of data mining. In this sense, time series can be easily obtained from financial and scientific applications, being one of the main sources of temporal datasets. How to discover useful time series patterns is a very interesting field [1]. The continuous nature of time series data make them difficult to be processed, analyzed and/or mined. Discretizing a continuous time series into significant symbols [2,3] is an option to alleviate this problem. The

* This work has been partially subsidised by the TIN2011-22794 project of the Spanish Ministry of Economy and Competitiveness (MINECO), FEDER funds and the P2011-TIC-7508 project of the "Junta de Andalucía" (Spain).
** Corresponding author at: Tel.: +34 957 218 349; Fax: +34 957 218 630.

Massive Missing Data Reconstruction in Ocean Buoys with Evolutionary Product Unit Neural Networks

A. M. Durán-Rosal[a,b], C. Hervás[a], A.J. Tallón-Ballesteros[c], A.C. Martínez-Estudillo[d], S. Salcedo-Sanz[e]

[a] *Department of Computer Science and Numerical Analysis, Universidad de Córdoba, Rabanales Campus, 14071 Córdoba, Spain.*

[b] *Corresponding author: Antonio M. Durán-Rosal. Department of Computer Science and Numerical Analysis, Universidad de Córdoba. 14071 Córdoba, Spain. Ph: +34 957218349, email: i92duroa@uco.es*

[c] *Department of Languages and Computer Systems, Universidad de Sevilla, 41012 Seville, Spain.*

[d] *Department of Management and Quantitative Methods, Universidad Loyola Andalucía, 41014 Seville, Spain.*

[e] *Department of Signal Processing and Communications, Universidad de Alcalá, 28805 Alcalá de Henares, Madrid, Spain.*

Abstract

In this paper we tackle the problem of massive missing data reconstruction in ocean buoys, with a Evolutionary Product Unit Neural Network (EPUNN). When considering a large number of buoys to reconstruct missing data, it is sometimes difficult to find a common period of completeness (without missing data on it) in the data to form a proper training and test set. In this paper we solve this issue by using partial reconstruction, which are then used as inputs of the EPUNN, with linear models. Missing data reconstruction in several phases or steps is then proposed. In this work we also show the potential of EPUNN to obtain simple, interpretable models in spite of the non-linear characteristic of the network, much simpler than the commonly used sigmoid-based neural systems. In the experimental section of the paper we show the performance of the proposed approach in a real case of massive missing data reconstruction in 6 wave-rider buoys at the Gulf of Alaska.

Keywords: Significant wave height; Missing values reconstruction; Product Unit Neural Networks; Evolutionary Algorithm.

On the use of evolutionary time series analysis for segmenting paleoclimate data

M. Pérez-Ortiz[1], A. Durán-Rosal[b], P.A. Gutiérrez[b], J. Sánchez-Monedero[1], A. Nikolaou[c], F. Fernández-Navarro[1], C. Hervás-Martínez[b]

[a] *Universidad Loyola Andalucía, Dpt. of Mathematics and Engineering, Third Building, 14004 Córdoba, Spain*
[b] *University of Córdoba, Dpt. of Computer Science and Numerical Analysis, Rabanales Campus, Albert Einstein building, 14071 Córdoba, Spain*
[c] *Institute of Planetary Research, German Aerospace Center, Berlin, Germany*

Abstract

Recent studies propose that some dynamical systems, such as climate, ecological and financial systems, among others, present critical transition points named to as *tipping points* (TPs). Climate TPs can severely affect millions of lives on Earth so that an active scientific community is working on finding early warning signals. This paper deals with the development of a time series segmentation algorithm for paleoclimate data in order to find segments sharing common statistical patterns. The proposed algorithm uses a clustering-based approach for evaluating the solutions and six statistical features, most of which have been previously considered in the detection of early warning signals in paleoclimate TPs. Due to the limitations of classical statistical methods, we propose the use of a genetic algorithm to automatically segment the series, together with a method to compare the segmentations. The final segments provided by the algorithm are used to construct a prediction model, whose promising results show the importance of segmentation for improving the understanding of a time series.

Key words: Time series segmentation, genetic algorithms, clustering, paleoclimate data, tipping points, abrupt climate change

In contrast to the famous statement of Linnaeus (1751) "*natura non facit saltus*" (or nature makes no leaps), it has been proven that some points of no return, thresholds and phase changes are widespread in nature and these are often non linear [1]. Such events can be rarely anticipated and some of them can have detrimental consequences on Earth's climate and large-scale impacts on human and ecological systems. Therefore, this increases the imperious necessity of studying, analysing and developing techniques for characterising them in order to construct reliable early warning systems. Although the human being have influenced their local environment for millennia, e.g. reducing biodiversity, it is now, since the industrial revolution, that truly global changes are being noticed [2, 3]. Examples that are currently receiving attention include the potential collapse of the Atlantic thermohaline circulation, the dieback of the Amazon rainforest or the decay of the Greenland ice sheet [1]. Formally, a climate "tipping point" (TP, also known as "little things can make a big difference") occurs when a small change in forcing triggers a strongly nonlinear response in the internal dynamics of part of the climate system, qualitatively changing its future state.

The critical relevance of early TPs detection has produced a growing attention of the scientific community. Lenton differences between several types of TPs, and presents some indicators that can help to detect them, such as the increase of autocorrelation of the series values [4]. In [5], more concrete techniques regarding data processing and indicators are presented. They study a bank of methods using only simulated ecological data concluding, in concordance with the literature, that there is no unique best indicator for identifying an upcoming transition. They also conclude that all the methods require specific data-treatment. Up to our knowledge, all previous works tackle the TP detection with statistical methods trying to select (by trial and error) the method more suitable to detect those transitions. They require an intensive data preprocessing that include, for instance, the use of Gaussian filters or rolling windows that introduces extra parameters (such as the width of the Gaussian function or size of the window) that need to be optimised [4, 5]. The main limitation behind these methods is that different TPs and different statistical descriptors require different and specific treatments.

Time series segmentation is a research field, aiming to provide a compact representation of the time series values, dividing it into segments and using an abstract representation of each segment. It is very important for time series representation and time series mining [6, 7] and is commonly used as a pre-processing step for different mining tasks [8] (e.g. clustering, classification or motif detection). In this way, segmentation algorithms have been used in many different fields, such as paleoecological problems [9], phoneme recognition [10] or paleontological climate [11]. Some recent works have proposed the use of algebraic segmentation for the specific case of short-term [12] and short nonstationary time series [13].

In the context of time series segmentation, this paper deals with climate time series segmentation. We introduce an evolu-

*This paper has been invited to be included in the "Special Issue Neurocomputing-HAIS2014".

Noname manuscript No.
(will be inserted by the editor)

Identifying market behaviours using European Stock Index time series by a hybrid segmentation algorithm

A.M. Durán-Rosal[1] · **M. de la Paz-Marín**[2] · **P.A. Gutiérrez**[1] · **C. Hervás-Martínez**[1]

the date of receipt and acceptance should be inserted later

Abstract The discovery of useful patterns embodied in a time series is of fundamental relevance in many real applications. Repetitive structures and common type of segments can also provide very useful information of patterns in financial time series. In this paper, we introduce a time series segmentation and characterization methodology combining a hybrid genetic algorithm and a clustering technique to automatically group common patterns from this kind of financial time series and address the problem of identifying stock market prices trends. This hybrid genetic algorithm includes a local search method aimed to improve the quality of the final solution. The local search algorithm is based on maximizing a likelihood ratio, assuming normality for the series and the subseries in which the original one is segmented. To do so, we select two stock market index time series: IBEX35 Spanish index (closing prices) and a weighted average (AVG) time series of the IBEX35 (Spanish), BEL20 (Belgian), CAC40 (French) and DAX (German) indexes. These are processed to obtain segments that are mapped into a five dimensional space composed of five statistical measures, with the purpose of grouping them according to their statistical properties. Experimental results show that it is possible to discover homogeneous patterns in both time series.

Keywords Time series segmentation, hybrid algorithms, clustering, European Stock Market Indexes.

1 Introduction

Recently, there has been a growing interest in mining time series databases [1]. Time series are an important class of temporal data objects that are collected and

Corresponding author at: Tel.: +34 957 218 349; Fax: +34 957 218 630.

University of Córdoba, Dept. of Computer Science and Numerical Analysis, Rabanales Campus, Albert Einstein building, 14071 - Córdoba, Spain E-mail: {i92duroa,pagutierrez,chervas}@uco.es · Department of Management and Quantitative Methods, Loyola Andalucía University, Business Administration Faculty, Escritor Castilla Aguayo 4, 14004 Córdoba, Spain E-mail: {mpaz}@uco.es


JUNTA DE ANDALUCIA


GOBIERNO
DE ESPAÑA
MINISTERIO
DE EDUCACIÓN, CULTURA
Y DEPORTE